EVAL80960VH Evaluation Platform

Board Manual

December 1998
# Contents

## 1 Introduction
1.1 EVAL80960VH Features ................................................................. 1-2
1.2 CTOOLS Software Development Toolset ....................................... 1-2
  1.2.1 CTOOLS and the MON960 Debug Monitor ......................... 1-2
    1.2.1.1 MON960 Host Communications ............................... 1-3
    1.2.1.2 Terminal Emulation Mode ........................................ 1-3
    1.2.1.3 Host Debugger Interface Library (HDI) Mode ............... 1-3
1.3 About This Manual .................................................................... 1-3
1.4 Notational Conventions ............................................................ 1-4
1.5 Technical Support .................................................................... 1-4
  1.5.1 Intel Customer Electronic Mail Support ............................. 1-5
  1.5.2 Intel Customer Literature and Telephone Support ............... 1-5
  1.5.3 Related Information ......................................................... 1-5

## 2 Getting Started
2.1 Pre-installation Considerations ................................................... 2-1
  2.1.1 Software Development Tools ............................................. 2-1
2.2 Software Installation ................................................................ 2-1
  2.2.1 Installing Software Development Tools .............................. 2-1
2.3 Hardware installation ................................................................. 2-2
  2.3.1 Installing the EVAL80960VH Platforms in the Host System .... 2-2
  2.3.2 Verify EVAL80960VH Platform is Functional ..................... 2-2
2.4 Creating and Downloading Executable Files ................................ 2-2
  2.4.1 Sample Download and Execution Using gdb960 ................. 2-2

## 3 Hardware Reference
3.1 Connectors, Switches and LEDs ................................................ 3-1
3.2 Power Requirements ................................................................ 3-2
3.3 DRAM ...................................................................................... 3-2
  3.3.1 DRAM Performance ......................................................... 3-2
  3.3.2 Upgrading DRAM ............................................................. 3-3
3.4 ROM and Flash ROM ............................................................... 3-3
3.5 Serial Port .............................................................................. 3-4
3.6 Logic Analyzer Headers ............................................................ 3-4
3.7 Expansion Connectors ............................................................... 3-6
3.8 JTAG Header ......................................................................... 3-8
3.9 Switch Settings ....................................................................... 3-8
3.10 User LEDs ............................................................................ 3-9
  3.10.1 User LEDs During Initialization ....................................... 3-10
3.11 Serial EEPROM (I²C) .............................................................. 3-10

## 4 i960® VH Processor Overview
4.1 CPU Memory Map ................................................................. 4-1
  4.1.1 Booting Out of DRAM on the EVAL80960VH Platform ...... 4-3
4.2 Local Interrupts ....................................................................... 4-3
4.3 CPU Counter/Timers ................................................................. 4-5
### 4.4 PCI Interface ................................................................. 4-5
### 4.5 DMA Channels ............................................................. 4-5

#### 5 MON960 Support for EVAL80960VH

##### 5.1 MON960 Components .................................................. 5-1
   - 5.1.1 mon960 Initialization ................................................ 5-1
   - 5.1.2 80960 JT Core Initialization ...................................... 5-1
   - 5.1.3 Memory Controller Initialization ................................. 5-2
   - 5.1.4 PCI Interface Initialization ...................................... 5-2
     - 5.1.4.1 Mode 0 ................................................................ 5-2
     - 5.1.4.2 Mode 1 ................................................................ 5-3
     - 5.1.4.3 Mode 2 ................................................................ 5-3
   - 5.1.5 ATU Initialization ..................................................... 5-3

##### 5.2 mon960 Kernel ............................................................. 5-4

##### 5.3 Diagnostics ............................................................... 5-4
   - 5.3.1 Board Level Diagnostics ............................................. 5-4

#### A Bill of Materials .............................................................. A-1

#### B Schematics ................................................................ B-1

#### C PLD Code .................................................................. C-1

#### D Backplane 0015 Interface ................................................ D-1
   - D.1 Introduction ................................................................. D-1
   - D.2 Installing the EVAL80960VH into the Backplane 0015 ........ D-1
   - D.3 Powering the EVAL80960VH/Backplane System .................. D-1
   - D.4 Interrupt Routing and IDSELs on the Backplane 0015 .......... D-2
   - D.5 PCI Host Detection and Configuration .............................. D-4
   - D.6 PCI Initialization ........................................................... D-4
   - D.7 PCI Bios Routines .......................................................... D-4
   - D.8 Additional MON960 Commands ..................................... D-9
   - D.9 Bill of Material ............................................................. D-9
   - D.10 Schematics ................................................................. D-9
   - D.11 PAL Code ................................................................. D-16

#### Figures

- 1-1 EVAL80960VH Platform Functional Block Diagram .................. 1-1
- 3-1 EVAL80960VH Platform Physical Diagram ............................... 3-1
- 3-2 Expansion Connector Pin Numbering ..................................... 3-6
- 3-3 LED Register Bitmap ......................................................... 3-9
- 4-1 i960® VH Processor Block Diagram ..................................... 4-1
- 4-2 EVAL80960VH Platform Memory Map .................................. 4-2
- 4-3 Interrupt Controller Connections ......................................... 4-4
- 4-4 i960® VH Processor DMA Controller ................................... 4-5
- D-1 EVAL80960VH Backplane Board Illustration .......................... D-2
- D-2 EVAL80960VH/Backplane 0015 with Eval Board Installed .......... D-3
- D-3 EVAL80960VH/Backplane 0015 Installation ........................... D-3
Tables

3-1 EVAL80960VH Connectors and LEDs ..............................................................3-1
3-2 EVAL80960VH Platform Power Requirements (Clock Mode DX4) ....................3-2
3-3 DRAM Performance ..........................................................................................3-3
3-4 DRAM Configurations .........................................................................................3-3
3-5 ROM Select .........................................................................................................3-3
3-6 UART Register Addresses ..................................................................................3-4
3-7 Logic Analyzer Header Definitions .................................................................3-4
3-8 Expansion Connector Definitions .......................................................................3-6
3-9 JTAG Header Pinout, Connector J2 ................................................................2-8
3-10 Switch S1 Settings .............................................................................................3-8
3-11 Startup LEDs MON960.....................................................................................3-10
4-1 DRAM Mapping ..................................................................................................4-3
4-2 DBAR Settings for Booting out of DRAM ...........................................................4-3
5-1 Initialization Modes .............................................................................................5-2
A-1 EVAL80960VH Bill of Materials .......................................................................A-1
D-1 PCI Interrupt Routing and IDSELs .................................................................D-2
D-2 Backplane 0015 Interface Bill of Material ........................................................D-9
This manual describes the EVAL80960VH evaluation platform for Intel’s i960® VH embedded-PCI processor. The i960 VH processor ("80960VH") addresses the needs of I/O processor applications by enabling communications, through message passing and interrupt generation, between it and a host processor or external PCI device. The 80960VH combines an 80960JT core with a PCI bus interface, as well as a memory controller, DMA channels, an interrupt controller interface, and an I2C Serial Bus. The EVAL80960VH evaluation platform is a half-length PCI adapter board. The board can be installed in any PCI host system that complies with the *PCI Local Bus Specification*, revision 2.1.

**Figure 1-1. EVAL80960VH Platform Functional Block Diagram**
Introduction

1.1 EVAL80960VH Features

The i960® VH embedded-PCI processor serves as the main component of a high performance, PCI-based I/O subsystem. The features of the EVAL80960VH platform are enumerated below and shown in Figure 1-1 and Figure 3-1.

- i960® VH embedded-PCI processor
- PCI short-card form factor
- 32-bit primary PCI bus interface
- 4 Mbytes DRAM expandable to 32Mbytes using 72-pin, 5V SIMM socket supporting EDO DRAM organized x32 or x36
- 2 DMA channels on PCI bus
- I²C Serial Bus
- 1K byte serial EEPROM (24C08) (connected to I²C bus for developer application use)
- Serial console port based on 16C550 UART
- Eight user-programmable LEDs
- 2 Indicator LEDs: processor has failed self-test, and processor is running
- Socketed flash ROM and on-board flash ROM devices
- Logic analyzer connectors for ROM bus and processor interface signals
- Processor local bus expansion header
- SMT dip switch for setting processor configuration signal defaults
- JTAG header (supports JTAG debugger interface)

1.2 CTOOLS Software Development Toolset

Intel’s i960 processor software development toolset (CTOOLS) features advanced C/C++-language compilers for the i960 processor family. CTOOLS development toolset is available for Windows® 95/NT-based systems and a variety of UNIX workstation hosts. These products provide execution profiling and instruction scheduling optimizations. The tools also include an assembler, linker, symbolic debugger, and utilities designed for embedded processor software development.

1.2.1 CTOOLS and the MON960 Debug Monitor

The EVAL80960VH platform is equipped with Intel’s mon960, an on-board software monitor that allows you to execute and debug programs written for i960 processors. The monitor provides program download, breakpoint, single step, memory display, and other useful functions for running and debugging a program.

The EVAL80960VH platform works with the source-level debuggers provided with CTOOLS, including gdb960 (command line version) and gdb960V (GUI version).
1.2.1.1 MON960 Host Communications

In a host system, mon960 allows you to communicate and download programs developed for the EVAL80960VH platform using the serial port, the PCI interface, or the JTAG interface. The EVAL80960VH platform supports two modes of serial connection: terminal emulation and Host Debugger Interface (HDI). The PCI and JTAG interfaces only support the HDI mode.

1.2.1.2 Terminal Emulation Mode

Terminal emulation software on your host system can communicate to mon960 on the EVAL80960VH platform via an RS-232 serial port. The EVAL80960VH platform supports port speeds from 300 to 115,200 bps. Downloading program code in this mode requires that the terminal emulation software support the XMODEM protocol.

Configure the serial port on the host system for 300-115,200 baud, 8 bits, one stop bit, no parity with XON/XOFF flow control. Six carriage returns must be sent to initiate this mode.

1.2.1.3 Host Debugger Interface Library (HDI) Mode

You may use a source-level debugger, such as Intel's gdb960 and gdb960V to establish serial or PCI, or JTAG communications with the EVAL80960VH platform. The mon960 Host Debugger Interface (HDI) provides a defined messaging layer between mon960 and the debugger. For more information on this interface, see the MON960 Debug Monitor User’s Guide (Intel order number 484290).

HDI connection requests cannot be detected by mon960 if the user has already initiated a connection using a terminal emulator, or vice versa. Once one mode is initiated, the EVAL80960VH platform must be reset before a different mode can be selected.

1.3 About This Manual

A brief description of the contents of this manual follows.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, &quot;Introduction&quot;</td>
<td>Introduces the EVAL80960VH evaluation board features. This chapter also describes Intel’s CTOOLS software development tools, and defines notational conventions and related documentation.</td>
</tr>
<tr>
<td>2, “Getting Started”</td>
<td>Provides step-by-step instructions for installing the EVAL80960VH platform in a host system and downloading and executing an application program. This chapter also describes Intel’s software development tools, the MON960 Debug Monitor, software installation, and hardware configuration.</td>
</tr>
<tr>
<td>3, “Hardware Reference”</td>
<td>Describes the locations of connectors, switches and LEDs on the EVAL80960VH. Header pinouts and register descriptions are also provided in this chapter.</td>
</tr>
<tr>
<td>4, “i960® VH Processor Overview”</td>
<td>Presents an overview of the capabilities of the i960® VH embedded-PCI processor and includes the CPU memory map.</td>
</tr>
<tr>
<td>5, “MON960 Support for EVAL80960VH”</td>
<td>Describes a number of features added to MON960 to support application development on the i960® VH embedded-PCI processor.</td>
</tr>
<tr>
<td>Appendix A, “Bill of Materials”</td>
<td>Shows a complete parts list of the EVAL80960VH evaluation board.</td>
</tr>
</tbody>
</table>
# 1.4 Notational Conventions

The following notation conventions are consistent with other i960 VH processor documentation and general industry standards.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>In code examples the pound symbol (#) or an overbar is appended to a signal name to indicate that the signal is active at a low voltage.</td>
</tr>
<tr>
<td>Bold</td>
<td>Indicates user entry and/or commands. PLD signal names are in bold lowercase letters (for example, h_off, h_on).</td>
</tr>
<tr>
<td>Italic</td>
<td>Indicates a reference to related documents; also used to show emphasis.</td>
</tr>
<tr>
<td>Courier font</td>
<td>Indicates code examples and file directories and names.</td>
</tr>
<tr>
<td>Asterisk (*)</td>
<td>On non-Intel company and product names, a trailing asterisk indicates the item is a trademark or registered trademark. Such brands and names are the property of their respective owners.</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>In text, signal names are shown in uppercase. When several signals share a common name, each signal is represented by the signal name followed by a number; the group is represented by the signal name followed by a variable ( n ). In code examples, signal names are shown in the case required by the software development tool in use.</td>
</tr>
<tr>
<td>Designations for hexadecimal and binary numbers</td>
<td>In text instead of using subscripted “base” designators (for example, FF(_{16})) or leading “0x” (for example, 0xFF) hexadecimal numbers are represented by a string of hex digits followed by the letter H. A zero prefix is added to numbers that begin with A through F. (for example, FF is shown as 0FFH.) In examples of actual code, “0x” is used. Decimal and binary numbers are represented by their customary notations. (for example, 255 is a decimal number and 1111 1111 is a binary number. In some cases, the letter B is added to binary numbers for clarity.)</td>
</tr>
</tbody>
</table>

## 1.5 Technical Support

Up-to-date product and technical information is available electronically from:

- Intel’s World-Wide Web (WWW) Location: [http://www.intel.com](http://www.intel.com)

For technical assistance, electronic mail (e-mail) provides the fastest route to reach engineers specializing in EVAL80960VH issues at the email address: 960tools@intel.com. Posting messages on the Embedded Microprocessor Forum at [http://support.intel.com/newsgroups/](http://support.intel.com/newsgroups/) is also a direct route for EVAL80960VH technical assistance. See Section 1.5.1.
Within the United States and Canada you may contact the Intel Technical Support Hotline. See Section 1.5.2 for a list of customer support sources for the US and other geographical areas.

1.5.1 Intel Customer Electronic Mail Support

For direct support from engineers specializing in i960® microprocessor issues, send email in english to 960tools@intel.com.

Questions and other messages may be posted to the Embedded Microprocessor Forum at http://support.intel.com/newsgroups/.

1.5.2 Intel Customer Literature and Telephone Support

Contact Intel Corporation for technical assistance for the EVAL80960VH evaluation platform.

<table>
<thead>
<tr>
<th>Country</th>
<th>Literature</th>
<th>Customer Support Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>800-548-4725</td>
<td>800-628-8686</td>
</tr>
<tr>
<td>Canada</td>
<td>800-468-8118 or 303-297-7763</td>
<td>800-628-8686</td>
</tr>
<tr>
<td>Europe</td>
<td>Contact local distributor</td>
<td>Contact local distributor</td>
</tr>
<tr>
<td>Australia</td>
<td>Contact local distributor</td>
<td>Contact local distributor</td>
</tr>
<tr>
<td>Israel</td>
<td>Contact local distributor</td>
<td>Contact local distributor</td>
</tr>
<tr>
<td>Japan</td>
<td>Contact local distributor</td>
<td>Contact local distributor</td>
</tr>
</tbody>
</table>

1.5.3 Related Information

To order printed manuals from Intel, contact your local sales representative or Intel Literature Sales in the US at 1-800-548-4725.

<table>
<thead>
<tr>
<th>Product</th>
<th>Document Name</th>
<th>Company/Order #</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Developers’ Insight CD-ROM</td>
<td>Intel # 273000</td>
</tr>
<tr>
<td>80960VH</td>
<td>i960® VH Processor Developer’s Manual</td>
<td>Intel # 273173</td>
</tr>
<tr>
<td>80960VH</td>
<td>i960® VH Processor Data Sheet</td>
<td>Intel # 273179</td>
</tr>
<tr>
<td>MON960</td>
<td>MON960 Debug Monitor User’s Guide</td>
<td>Intel #484290</td>
</tr>
<tr>
<td></td>
<td>PCI Local Bus Specification Revision 2.1</td>
<td>PCI Special Interest Group 1-800-433-5177</td>
</tr>
<tr>
<td>80960JX</td>
<td>i960® Jx I/O Microprocessor User’s Manual</td>
<td>Intel # 272483</td>
</tr>
</tbody>
</table>

Contact Cyclone Microsystems for additional information about their products and literature:

<table>
<thead>
<tr>
<th>Cyclone Microsystems</th>
<th>Phone:203-786-5536</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 Science Park</td>
<td>FAX:203-786-5025</td>
</tr>
<tr>
<td>New Haven CT 06511</td>
<td>e-mail:<a href="mailto:info@cyclone.com">info@cyclone.com</a></td>
</tr>
<tr>
<td></td>
<td>WWW:<a href="http://www.cyclone.com">http://www.cyclone.com</a></td>
</tr>
</tbody>
</table>
This chapter contains instructions for installing the EVAL80960VH platform in a host system, and downloading and executing an application program using Intel’s CTOOLS software development toolsets.

2.1 Pre-installation Considerations

This section provides a general overview of the components required to develop and execute a program on the EVAL80960VH platform. The *MON960 Debug Monitor User’s Guide* (Intel order number 484290) describes several of these components, including mon960 commands, the Host Debugger Interface Library (HDIL), and the mondb.exe utility.

2.1.1 Software Development Tools

A number of software development tools are available for the i960® processor family. The installation instructions presented in this chapter were verified using CTOOLS - Intel’s i960 processor software development tools. CTOOLS is a complete C/C++ - language software development toolset for developing embedded applications to run on i960 processors. It contains a C/C++ compiler, the gcc960 and ic960 compiler driver programs, an assembler, linker, symbolic debugger, runtime libraries, a collection of software development tools and utilities, and printed and on-line documentation. The MON960 Debug Monitor User’s Guide fully describes the components of mon960, including mon960 commands, the Host Debugger Interface Library (HDIL), and the mondb.exe utility. If you are using mon960 and the CTOOLS toolset, then refer to Section 2.2.1, “Installing Software Development Tools” on page 2-1. If you are using other software development tools, then read through this example to gain a general understanding of how to use tools with this board.

2.2 Software Installation

2.2.1 Installing Software Development Tools

Install your development software as described in its manuals. All references in this manual to CTOOLS assume that the default directories were selected during installation. If this is not the case, then substitute the appropriate path for the default path wherever file locations are referenced in this manual.

---

1. Refer to Intel’s http://developer.intel.com web page catalog for a complete list of i960 processor software development and debug tools.
2.3 Hardware installation

Follow these instructions to get your new EVAL80960VH platform running. Be sure all items on the checklist provided in your kit’s customer letter were provided with your EVAL80960VH evaluation platform kit.

Warning: Static charges can severely damage the EVAL80960VH platforms. Be sure you are properly grounded before removing the EVAL80960VH evaluation board from the anti-static bag.

2.3.1 Installing the EVAL80960VH Platforms in the Host System

If you are installing the EVAL80960VH platform for the first time, then visually inspect the board for any damage that may have occurred during shipping. If there are visible defects, then return the board for replacement. Follow the host system manufacturer’s instructions for installing a PCI adapter. The EVAL80960VH platform is a half-length PCI adapter and requires a PCI slot that is free of obstructions.

2.3.2 Verify EVAL80960VH Platform is Functional

These instructions assume that you have already installed the EVAL80960VH platform in the host system as described in Section 2.3.1.

1. To connect the serial port for communicating with and downloading to the EVAL80960VH platform, connect the RS-232 cable (provided with the EVAL80960VH kit) from a free serial port on a host system to the phone jack-style connector on the EVAL80960VH platform.

2. After power-up, the red FAIL LED should turn off, indicating that the processor has passed its self-test.

3. Press <return> six times on a terminal connected to the EVAL80960VH platform to bring up the mon960 prompt. mon960 automatically adjusts its baud rate to match that of the terminal at start-up.

2.4 Creating and Downloading Executable Files

To download code to the EVAL80960VH platform, your compiler should produce an ELF-format object file. Consult the CTOOLS documentation for information regarding compiling, linking, and downloading applications.

During a download, mon960 checks the link address stored in the ELF file, and stores the file at that location on the EVAL80960VH platform. If the executable file is linked to an invalid address, then mon960 will abort the download.

2.4.1 Sample Download and Execution Using gdb960

This example shows you how to use gdb960 to download and execute a file named myapp via the serial port.

- Invoke gdb960. For example, from a Windows* 95/NT or UNIX command prompt, issue the command:
gdb960 -r com2 myapp
This command establishes communication and downloads the file myapp.

- To execute the program, enter the command from the gdb960 command prompt:
  (gdb960) run

More information on the gdb960 commands mentioned in this section can be found in the GDB960 User’s Manual.
This section describes the location and function of physical connectors, switches and LEDs. Refer to Figure 3-1, the physical diagram of the EVAL80960VH platform, for the locations of components discussed in this section.

3.1 Connectors, Switches and LEDs

Figure 3-1 shows the physical locations of the major components on the EVAL80960VH platform. Table 3-1 lists the functions of these components. For a complete list of components on the EVAL80960VH platform, refer to Appendix A, “Bill of Materials”.

Table 3-1. EVAL80960VH Connectors and LEDs (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J9, J10</td>
<td>PCI bus card edge connector</td>
</tr>
<tr>
<td>J2</td>
<td>JTAG connector</td>
</tr>
<tr>
<td>J1</td>
<td>Serial port connector</td>
</tr>
</tbody>
</table>
3.2 Power Requirements

The EVAL80960VH platform draws power from the PCI bus. The power requirements of the EVAL80960VH platform are shown in Table 3-2.

Table 3-2. EVAL80960VH Platform Power Requirements (Clock Mode DX4)

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Typical Current</th>
<th>Maximum Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>+3.3 V</td>
<td>0 V*</td>
<td>0 V*</td>
</tr>
<tr>
<td>+5 V</td>
<td>.985 A</td>
<td>1.23 A</td>
</tr>
<tr>
<td>+12 V</td>
<td>50 mA</td>
<td>50 mA</td>
</tr>
<tr>
<td>-12 V</td>
<td>10 mA</td>
<td>10 mA</td>
</tr>
</tbody>
</table>

* +3.3V for the 80960VH created on board from +5V

3.3 DRAM

The EVAL80960VH platform is equipped with 4 Mbytes of DRAM. This SIMM module can be upgraded to an 8, 16, or 32 Mbyte module.

3.3.1 DRAM Performance

The EVAL80960VH platform uses extended data out (EDO) DRAM to achieve a zero-wait-state burst at 33 MHz. The memory runs with two wait states in the first cycle of a read burst and one wait state during a write burst. There are no wait states during the burst. No additional recovery cycles are required beyond the intrinsic i960 JT core recovery cycle. Table 3-3 shows the performance numbers for the EVAL80960VH platform.
Note: Bandwidth is sustained bandwidth, not peak.

### Table 3-3. DRAM Performance

<table>
<thead>
<tr>
<th>Cycle Type</th>
<th>Table Clocks</th>
<th>Wait States</th>
<th>Performance Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Single</td>
<td>4</td>
<td>2</td>
<td>27 Mbytes/sec</td>
</tr>
<tr>
<td>Read Burst</td>
<td>4-1-1-1</td>
<td>2-0-0-0</td>
<td>66 Mbytes/sec</td>
</tr>
<tr>
<td>Write Single</td>
<td>3</td>
<td>1</td>
<td>33 Mbytes/sec</td>
</tr>
<tr>
<td>Write Burst</td>
<td>3-1-1-1</td>
<td>1-0-0-0</td>
<td>76 Mbytes/sec</td>
</tr>
</tbody>
</table>

### 3.3.2 Upgrading DRAM

The EVAL80960VH is equipped with 4 Mbytes of DRAM in the SIMM socket. It may be expanded by changing the module from 4 up to 32 Mbytes of DRAM. The various memory combinations are shown in Table 3-4. Only 72 pin EDO modules rated at 60 ns should be used on the platform. Either x32 or x36 devices may be used.

### Table 3-4. DRAM Configurations

<table>
<thead>
<tr>
<th>Total Memory</th>
<th>SIMM Module Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mbyte</td>
<td>1M x 32 or 1M x 36 (4 Mbytes)</td>
</tr>
<tr>
<td>8 Mbytes</td>
<td>2M x 32 or 2M x 36</td>
</tr>
<tr>
<td>16 Mbytes</td>
<td>4M x 32 or 4M x 36</td>
</tr>
<tr>
<td>32 Mbyte</td>
<td>8M x 32 or 8M x 36</td>
</tr>
</tbody>
</table>

### 3.4 ROM and Flash ROM

A 32-pin PLCC ROM socket is included on the EVAL80960VH platform. The ROM socket at U13 is populated with a 28F020 Flash ROM containing mon960. The ROM at U14 is a 28F016S5 Flash ROM and may be used to store user applications. mon960 includes features to erase and program the 28F016S5 Flash ROM and download code directly into Flash ROM. The Flash ROM at U13 cannot be programmed on-board. It must be removed from the board and programmed using a Flash programmer. Table 3-5 shows the switch settings at switch S2 to select which ROM to boot from.

### Table 3-5. ROM Select

<table>
<thead>
<tr>
<th>ROMSWAP (S2-1)</th>
<th>ROMDISABLE (S2-2)</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>Boot from U14 SMT ROM</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>Boot from U13 PLCC ROM or ROM emulator inserted into PLCC socket</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>Boot from MICTOR connector populated with a ROM emulator</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>Boot from MICTOR connector populated with a ROM emulator</td>
</tr>
</tbody>
</table>
3.5 Serial Port

The serial port on the EVAL80960VH platform, based on a 16C550 UART, is capable of operation from 300 to 115,200 bps. The port is connected to a phone jack-style plug on the EVAL80960VH platform. The DB25 to RJ-45 cable included with the EVAL80960VH can be used to connect the console port to any standard RS-232 port on the host system.

The UART on the EVAL80960VH platform is clocked with a 1.843 MHz clock and may be programmed to use this clock with its internal baud rate counters. The UART register addresses are shown in Table 3-6. Refer to the 16C550 device data book for a detailed description of the registers and device operation. Note that some UART addresses refer to different registers depending on whether a read or a write is being performed.

Table 3-6. UART Register Addresses

<table>
<thead>
<tr>
<th>Address</th>
<th>Read Register</th>
<th>Write Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>E000 0000H</td>
<td>Receive Holding Register</td>
<td>Transmit Holding Register</td>
</tr>
<tr>
<td>E000 0004H</td>
<td>Unused</td>
<td>Interrupt Enable Register</td>
</tr>
<tr>
<td>E000 0008H</td>
<td>Interrupt Status Register</td>
<td>FIFO Control Register</td>
</tr>
<tr>
<td>E000 000CH</td>
<td>Unused</td>
<td>Line Control Register</td>
</tr>
<tr>
<td>E000 0010H</td>
<td>Unused</td>
<td>Modem Control Register</td>
</tr>
<tr>
<td>E000 0014H</td>
<td>Line Status Register</td>
<td>Unused</td>
</tr>
<tr>
<td>E000 0018H</td>
<td>Modern Status Register</td>
<td>Unused</td>
</tr>
<tr>
<td>E000 001CH</td>
<td>Scratchpad Register</td>
<td>Scratchpad Register</td>
</tr>
</tbody>
</table>

3.6 Logic Analyzer Headers

There are three logic analyzer connectors on the EVAL80960VH platform. The connectors are Mictor type, AMP part # 767054-1. Hewlet-Packard and Tektronix manufacture and sell interfaces to these connectors. The logic analyzer connectors allow for interfacing to the ROM bus along with processor interface signals. Table 3-7 shows the connectors and the pin assignments for each.

Table 3-7. Logic Analyzer Header Definitions (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>PIN</th>
<th>TEK</th>
<th>HP</th>
<th>CONN A</th>
<th>CONN B</th>
<th>CONN C</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>CLK</td>
<td>CLK</td>
<td>CLK_MIC2</td>
<td>CLK_MIC1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3:7</td>
<td>D15</td>
<td>ROMA15</td>
<td>AD15</td>
<td>LRST#</td>
</tr>
<tr>
<td>5</td>
<td>3:6</td>
<td>D14</td>
<td>ROMA14</td>
<td>AD14</td>
<td>LOCK#</td>
</tr>
<tr>
<td>6</td>
<td>3:5</td>
<td>D13</td>
<td>MA11</td>
<td>AD13</td>
<td>LRDYRCV#</td>
</tr>
<tr>
<td>7</td>
<td>3:4</td>
<td>D12</td>
<td>MA10</td>
<td>AD12</td>
<td>RDYRCV#</td>
</tr>
<tr>
<td>8</td>
<td>3:3</td>
<td>D11</td>
<td>MA9</td>
<td>AD11</td>
<td>BLAST#</td>
</tr>
<tr>
<td>9</td>
<td>3:2</td>
<td>D10</td>
<td>MA8</td>
<td>AD10</td>
<td>DEN#</td>
</tr>
<tr>
<td>10</td>
<td>3:1</td>
<td>D9</td>
<td>MA7</td>
<td>AD9</td>
<td>DT/R</td>
</tr>
<tr>
<td>11</td>
<td>3:0</td>
<td>D8</td>
<td>MA6</td>
<td>AD8</td>
<td>W_RD#</td>
</tr>
</tbody>
</table>
Table 3-7. Logic Analyzer Header Definitions (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>PIN</th>
<th>TEK.</th>
<th>HP</th>
<th>CONN A</th>
<th>CONN B</th>
<th>CONN C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>J6</td>
<td>J4</td>
<td>J5</td>
</tr>
<tr>
<td>12</td>
<td>2:7</td>
<td>D7</td>
<td>MA5</td>
<td>AD7</td>
<td>D/C</td>
</tr>
<tr>
<td>13</td>
<td>2:6</td>
<td>D6</td>
<td>MA4</td>
<td>AD6</td>
<td>WIDTH1</td>
</tr>
<tr>
<td>14</td>
<td>2:5</td>
<td>D5</td>
<td>MA3</td>
<td>AD5</td>
<td>WIDTH0</td>
</tr>
<tr>
<td>15</td>
<td>2:4</td>
<td>D4</td>
<td>MA2</td>
<td>AD4</td>
<td>ADS#</td>
</tr>
<tr>
<td>16</td>
<td>2:3</td>
<td>D3</td>
<td>MA1</td>
<td>AD3</td>
<td>BE3#</td>
</tr>
<tr>
<td>17</td>
<td>2:2</td>
<td>D2</td>
<td>MA0</td>
<td>AD2</td>
<td>BE1#</td>
</tr>
<tr>
<td>18</td>
<td>2:1</td>
<td>D1</td>
<td>BE1#</td>
<td>AD1</td>
<td>BE1#</td>
</tr>
<tr>
<td>19</td>
<td>2:0</td>
<td>D0</td>
<td>BE#2</td>
<td>AD0</td>
<td>BE0#</td>
</tr>
<tr>
<td>20</td>
<td>0:0</td>
<td>D0</td>
<td>ROMA16</td>
<td>AD16</td>
<td>XINT0#</td>
</tr>
<tr>
<td>21</td>
<td>0:1</td>
<td>D1</td>
<td>ROMA17</td>
<td>AD17</td>
<td>XINT1#</td>
</tr>
<tr>
<td>22</td>
<td>0:2</td>
<td>D2</td>
<td>ROMA18</td>
<td>AD18</td>
<td>XINT2#</td>
</tr>
<tr>
<td>23</td>
<td>0:3</td>
<td>D3</td>
<td>ROMA19</td>
<td>AD19</td>
<td>XINT3#</td>
</tr>
<tr>
<td>24</td>
<td>0:4</td>
<td>D4</td>
<td>ROMA20</td>
<td>AD20</td>
<td>XINT4#</td>
</tr>
<tr>
<td>25</td>
<td>0:5</td>
<td>D5</td>
<td>AD0</td>
<td>AD21</td>
<td>XINT5#</td>
</tr>
<tr>
<td>26</td>
<td>0:6</td>
<td>D6</td>
<td>AD1</td>
<td>AD22</td>
<td>XINT6#</td>
</tr>
<tr>
<td>27</td>
<td>0:7</td>
<td>D7</td>
<td>AD2</td>
<td>AD23</td>
<td>XINT7#</td>
</tr>
<tr>
<td>28</td>
<td>1:0</td>
<td>D8</td>
<td>AD3</td>
<td>AD24</td>
<td>NMI#</td>
</tr>
<tr>
<td>29</td>
<td>1:1</td>
<td>D9</td>
<td>AD4</td>
<td>AD25</td>
<td>HOLD</td>
</tr>
<tr>
<td>30</td>
<td>1:2</td>
<td>D10</td>
<td>AD5</td>
<td>AD26</td>
<td>HOLDA</td>
</tr>
<tr>
<td>31</td>
<td>1:3</td>
<td>D11</td>
<td>AD6</td>
<td>AD27</td>
<td>DREQ#</td>
</tr>
<tr>
<td>32</td>
<td>1:4</td>
<td>D12</td>
<td>AD7</td>
<td>AD28</td>
<td>DACK#</td>
</tr>
<tr>
<td>33</td>
<td>1:5</td>
<td>D13</td>
<td>CE0#</td>
<td>AD29</td>
<td>WAIT#</td>
</tr>
<tr>
<td>34</td>
<td>1:6</td>
<td>D14</td>
<td>MWE0#</td>
<td>AD30</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1:7</td>
<td>D15</td>
<td>W_RD#</td>
<td>AD31</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>CLK</td>
<td>CLK</td>
<td>LRST</td>
<td>ALE</td>
</tr>
</tbody>
</table>
3.7 Expansion Connectors

Table 3-8 details the pinout of the expansion connector. The expansion connectors are located at the right end of the board, away from the end panel. This allows users to use the rest of the slot for custom I/O modules. The expansion connectors are standard 0.025” socket strips arranged 2 x 25 and numbered as shown Figure 3-2.

**Figure 3-2. Expansion Connector Pin Numbering**

![Expansion Connector Pin Numbering](image)

<table>
<thead>
<tr>
<th>PIN</th>
<th>CONN A</th>
<th>CONN B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BE0#</td>
<td>AD0</td>
</tr>
<tr>
<td>2</td>
<td>BE1#</td>
<td>AD1</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>AD2</td>
</tr>
<tr>
<td>4</td>
<td>BE3#</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>BE2#</td>
<td>AD3</td>
</tr>
<tr>
<td>6</td>
<td>GND</td>
<td>AD4</td>
</tr>
<tr>
<td>7</td>
<td>ADS#</td>
<td>AD5</td>
</tr>
<tr>
<td>8</td>
<td>WIDTH0</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>AD6</td>
</tr>
<tr>
<td>10</td>
<td>D/C</td>
<td>AD7</td>
</tr>
<tr>
<td>11</td>
<td>WIDTH1</td>
<td>AD8</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>13</td>
<td>W_RD#</td>
<td>AD9</td>
</tr>
<tr>
<td>14</td>
<td>DT/R</td>
<td>AD10</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>AD11</td>
</tr>
<tr>
<td>16</td>
<td>BLAST#</td>
<td>GND</td>
</tr>
</tbody>
</table>
Table 3-8. Expansion Connector Definitions  (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>PIN</th>
<th>CONN A</th>
<th>CONN B</th>
</tr>
</thead>
<tbody>
<tr>
<td>J7</td>
<td>J8</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>DEN#</td>
<td>AD12</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>AD13</td>
</tr>
<tr>
<td>19</td>
<td>RDYRCV#</td>
<td>AD14</td>
</tr>
<tr>
<td>20</td>
<td>LRDYRCV#</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>AD15</td>
</tr>
<tr>
<td>22</td>
<td>HOLD</td>
<td>AD16</td>
</tr>
<tr>
<td>23</td>
<td>LOCK#</td>
<td>AD17</td>
</tr>
<tr>
<td>24</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>HOLDA</td>
<td>AD18</td>
</tr>
<tr>
<td>26</td>
<td>XINT0#</td>
<td>AD19</td>
</tr>
<tr>
<td>27</td>
<td>GND</td>
<td>AD20</td>
</tr>
<tr>
<td>28</td>
<td>XINT2#</td>
<td>GND</td>
</tr>
<tr>
<td>29</td>
<td>XINT1#</td>
<td>AD21</td>
</tr>
<tr>
<td>30</td>
<td>GND</td>
<td>AD22</td>
</tr>
<tr>
<td>31</td>
<td>XINT3#</td>
<td>AD23</td>
</tr>
<tr>
<td>32</td>
<td>XINT4#</td>
<td>GND</td>
</tr>
<tr>
<td>33</td>
<td>GND</td>
<td>AD24</td>
</tr>
<tr>
<td>34</td>
<td>XINT6#</td>
<td>AD25</td>
</tr>
<tr>
<td>35</td>
<td>XINT5#</td>
<td>AD26</td>
</tr>
<tr>
<td>36</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>37</td>
<td>XINT7#</td>
<td>AD27</td>
</tr>
<tr>
<td>38</td>
<td>NMI#</td>
<td>AD28</td>
</tr>
<tr>
<td>39</td>
<td>GND</td>
<td>AD29</td>
</tr>
<tr>
<td>40</td>
<td>DACK#</td>
<td>GND</td>
</tr>
<tr>
<td>41</td>
<td>DREQ3</td>
<td>AD30</td>
</tr>
<tr>
<td>42</td>
<td>GND</td>
<td>AD31</td>
</tr>
<tr>
<td>43</td>
<td>WAIT#</td>
<td>GND</td>
</tr>
<tr>
<td>44</td>
<td>LRST#</td>
<td>GND</td>
</tr>
<tr>
<td>45</td>
<td>+5V</td>
<td>ALE</td>
</tr>
<tr>
<td>46</td>
<td>+5V</td>
<td>+5V</td>
</tr>
<tr>
<td>47</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>48</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>49</td>
<td>GND</td>
<td>+5V</td>
</tr>
<tr>
<td>50</td>
<td>GND</td>
<td>CLK_EXP</td>
</tr>
</tbody>
</table>
3.8 JTAG Header

The JTAG header allows debugging hardware to be quickly and easily connected to the EVAL80960VH evaluation platform.

The JTAG header is a 16 pin header. A 3M connector (Part Number 2516-6002UG) is required to connect to this header. The pinout for the JTAG header is shown in Table 3-9. The header and connector are keyed using a tab on the connector and a slot on the header to ensure proper installation.

Each signal in the JTAG header is paired with its own ground connection to avoid the noise problems associated with long ribbon cables. Signal descriptions are found in the i960® VH Processor Developer's Manual, and i960® VH Processor Data Sheet.

Table 3-9. JTAG Header Pinout, Connector J2

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
<th>INPUT/OUTPUT TO 80960VH</th>
<th>PIN</th>
<th>SIGNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TRST#</td>
<td>IN</td>
<td>2</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>TDI</td>
<td>IN</td>
<td>4</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>TDO</td>
<td>OUT</td>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>7</td>
<td>TMS</td>
<td>IN</td>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>TCK</td>
<td>IN</td>
<td>10</td>
<td>GND</td>
</tr>
<tr>
<td>11</td>
<td>LCDINIT#</td>
<td>IN</td>
<td>12</td>
<td>GND</td>
</tr>
<tr>
<td>13</td>
<td>I_RST#</td>
<td>OUT</td>
<td>14</td>
<td>GND</td>
</tr>
<tr>
<td>15</td>
<td>PWRVLD</td>
<td>OUT</td>
<td>16</td>
<td>GND</td>
</tr>
</tbody>
</table>

3.9 Switch Settings

Options for switch S1 settings are listed in Table 3-10. Refer to i960® VH Processor Data Sheet (order number 273179) for pin descriptions of these processor signals.

Table 3-10. Switch S1 Settings (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Position</th>
<th>Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1-1</td>
<td>RETRY (Active only when RST_MODE# is low.)</td>
<td>Determines if the Primary PCI interface will be disabled. ON = allows Primary PCI configuration cycles to occur. OFF = retries all Primary PCI configuration cycles.</td>
<td>OFF</td>
</tr>
<tr>
<td>S1-2</td>
<td>RST_MODE#</td>
<td>Determines if the processor is to be held in reset. ON = hold in reset. OFF = allows processor initialization (processor forces RETRY low).</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Hardware Reference

3.10 User LEDs

The EVAL80960VH platform has a bank of eight user-programmable LEDs, located on the upper edge of the adapter board. These LEDs are controlled by a write-only register and used as a debugging aid during development. Software can control the state of the user LEDs by writing to the LED Register, located at E004 0000H. Each of the eight bits of this register corresponds to one of the user LEDs. Clearing a bit in the LED Register by writing a “0” to it turns the corresponding LED on, while setting a bit by writing a “1” to it turns the corresponding LED off. Resetting the EVAL80960VH platform results in clearing the register and turning all the LEDs on. The LED Register bitmap is shown in Figure 3-3.

The user LEDs are numbered in descending order from left to right, with LED7 being on the left when looking at the component side of the adapter.

Table 3-10. Switch S1 Settings  (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Position</th>
<th>Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1-3*</td>
<td>CLKMODE0#</td>
<td>Clock MODE are used to select the mode of operation in terms of 80960 local bus/PCI bus vs. the internal 80960 processor.</td>
<td>OFF</td>
</tr>
<tr>
<td>S1-4*</td>
<td>CLKMODE1#</td>
<td>Clock MODE are used to select the mode of operation in terms of 80960 local bus/PCI bus vs. the internal 80960 processor.</td>
<td>OFF</td>
</tr>
</tbody>
</table>

*CLOCK MODES
00 - DX4 mode
01 - DX2 mode
10 - DX mode
11 - select speed using the clock mode bits in the CSR register

Figure 3-3. LED Register Bitmap
3.10.1 User LEDs During Initialization

MON960 indicates the progress of its hardware initialization on the user LEDs. In the event that initialization should fail for some reason, the number of the lit LED can be used to determine the cause of the failure. Table 3-11 lists the tests that correspond to each lit LED.

Table 3-11. Startup LEDs MON960

<table>
<thead>
<tr>
<th>LEDs</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED 0</td>
<td>UART walking ones test passed.</td>
</tr>
<tr>
<td>LED 1</td>
<td>DRAM walking ones test passed.</td>
</tr>
<tr>
<td>LED 2</td>
<td>DRAM multiword test passed.</td>
</tr>
<tr>
<td>LED 3</td>
<td>Memory controller initialized.</td>
</tr>
<tr>
<td>LED 4</td>
<td>Hardware initialization started.</td>
</tr>
<tr>
<td>LED 5</td>
<td>Flash ROM initialized.</td>
</tr>
<tr>
<td>LED 6</td>
<td>ATU initialized.</td>
</tr>
<tr>
<td>LED 7</td>
<td>UART internal loopback test passed.</td>
</tr>
</tbody>
</table>

3.11 Serial EEPROM (I²C)

A 1 Kbyte serial EEPROM is connected to the I²C bus on the EVAL80960VH platform at address 0. Intel does not define the contents of this device, so it is available for use by the developer. The EEPROM is read and written using the I²C bus; consult the i960® VH Processor Developer’s Manual and the 24C08 Serial EEPROM Data Sheet for more information.
This chapter describes the features and operation of the i960 VH embedded-PCI processor on the EVAL80960VH platform. For more detail, refer to the i960® VH Processor Developer’s Manual.

**Figure 4-1. i960® VH Processor Block Diagram**

### 4.1 CPU Memory Map

The memory map for the EVAL80960VH platform is shown in Figure 4-2. All addresses below 9002 0000H on the EVAL80960VH platform are reserved for various functions of the i960 VH embedded-PCI processor, as shown on the memory map. Documentation for these areas, as well as the processor memory mapped registers at FF00 0000H and the IBR, can be found in the i960® VH Processor Developer’s Manual.
Figure 4-2. EVAL80960VH Platform Memory Map

Flash
ROM, and
Processor Registers

On-board Devices

Reserved

DRAM

Reserved

ATU Outbound Translation
Windows

ATU Outbound Direct
Addressing Window

Reserved

Peripheral Memory Mapped
Registers

Reserved

Processor Internal Data RAM

Processor Memory Mapped Registers

ROM B*

ROM A*

Reserved

DRAM Bank Size Register (write only)

LED Register (write only)

UART

FF00 0000H
FEE0 0000H
FEC0 0000H
F000 0000H
E020 0000H
E004 0000H
E000 0000H
**Note:** With the DRAM base register set to A000 0000H the mapping for the DRAM, based on DRAM size, is shown in Table 4-1.

### Table 4-1. DRAM Mapping

<table>
<thead>
<tr>
<th>DRAM Size</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mbytes</td>
<td>A3C0 0000H - A3FF FFFFH</td>
</tr>
<tr>
<td>8 Mbytes</td>
<td>A380 0000H - A3FF FFFFH</td>
</tr>
<tr>
<td>16 Mbytes</td>
<td>A300 0000H - A3FF FFFFH</td>
</tr>
<tr>
<td>32 Mbytes</td>
<td>A200 0000H - A3FF FFFFH</td>
</tr>
</tbody>
</table>

### 4.1.1 Booting Out of DRAM on the EVAL80960VH Platform

The EVAL80960VH platform provides for the ability to map DRAM to allow booting out of DRAM. This can be accomplished with all DRAM configurations except with a 16 Mbyte module. The IBR is located on the i960 VH processor beginning at location FEFF FF30H. Table 4-2 shows what address the DRAM Base Address Register (DBAR 1520H) should be set to for a given memory module size. Note that addresses above FEFF FF5FH are reserved. If a 16 Mbyte module is used, the base address would be FC00 0000H and the range would be from FF00 0000H to FFFF FFFFH and the IBR would not be contained within the DRAM space. DRAM must be on an address boundary equal to the total size of 4 DRAM banks, in these cases 4M x 4 or 16M x 4.

### Table 4-2. DBAR Settings for Booting out of DRAM

<table>
<thead>
<tr>
<th>Memory Size</th>
<th>Base Address</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Mbytes</td>
<td>FE00 0000H</td>
<td>FE00 0000H - FEFF FFFFH</td>
</tr>
<tr>
<td>8 Mbytes</td>
<td>FE00 0000H</td>
<td>FE80 0000H - FEFF FFFFH</td>
</tr>
<tr>
<td>32 Mbytes</td>
<td>FC00 0000H</td>
<td>FE00 0000H - FFFF FFFFH</td>
</tr>
</tbody>
</table>

### 4.2 Local Interrupts

The i960 VH embedded-PCI processor is built around an 80960JT core, which has seven external interrupt lines designated XINT0# through XINT7# and NMI#. In the 80960VH processor, these interrupt lines are not directly connected to external interrupts, but pass through a layer of internal interrupt routing logic. Figure 4-3 shows the interrupt connections on the 80960VH processor.

On the EVAL80960VH platform, XINT5# is connected to the 16C550 UART. XINT6#, XINT7#, and NMI# can receive interrupts from internal sources or from external sources. Since all of these interrupts accept signals from multiple sources, a status register is provided for each of them to allow service routines to identify the source of the interrupt. Each of the possible interrupt sources is assigned a bit position in the status register. The interrupt sources for these lines are shown in Figure 4-3. On the EVAL80960VH platform, the NMI# interrupt is not connected to any external interrupt source and receives interrupts only from the internal devices on the i960 VH embedded-PCI processor. Note that error indications are received on NMI#.
Figure 4-3. Interrupt Controller Connections
4.3 CPU Counter/Timers

The i960 VH embedded-PCI processor is equipped with two 32 bit on-chip counter/timers which are clocked with the 80960VH processor clock signal. The 80960VH processor receives its clock from the primary PCI interface clock, generated by the motherboard. Most motherboards generate a 33 MHz clock signal, although the PCI specification only requires a clock frequency between 0 and 33 MHz. The timers can be programmed for single-shot or continuous mode, and can generate interrupts to the processor when the countdown expires.

4.4 PCI Interface

The PCI interface on the EVAL80960VH platform provides the i960 VH embedded-PCI processor with a connection to the PCI bus on the host system.

4.5 DMA Channels

The i960 VH embedded-PCI processor features two independent DMA channels, both of which operate on the PCI interface. Both of the DMA channels connect to the 80960VH processor’s local bus and can be used to transfer data from PCI devices to memory on the EVAL80960VH platform. Support for chaining, and scatter/gather is built into both channels. The DMA can address the entire $2^{64}$ bytes of address space on the PCI bus and $2^{32}$ bytes of address space on the internal bus.

Figure 4-4. i960® VH Processor DMA Controller
This chapter discusses a number of additions that have been made to MON960 to support the EVAL80960VH. For complete documentation on the operation of mon960, see the MON960 Debug Monitor User’s Guide. The EVAL80960VH evaluation platform ships with mon960 in flash firmware. See Chapter 3, “Hardware Reference” for more information on updating the onboard Flash. See Chapter 1, “Introduction” for a description of CTOOLS.

5.1 MON960 Components

The remaining sections of this chapter assume that mon960 is installed in the onboard Flash. The EVAL80960VH mon960 debug monitor consists of four main components:

• Initialization firmware
• mon960 kernel
• mon960 extensions
• Diagnostics/example code

These four components together are referred to as MON960.

5.1.1 mon960 Initialization

At initialization, mon960 puts the EVAL80960VH platform into a known, functional state that allows the host processor to perform PCI initialization. Once in this state, the mon960 kernel and the mon960 extensions can load and execute correctly. Initialization is performed after a RESET condition. mon960 initialization encompasses all major portions of the i960® VH embedded-PCI processor and EVAL80960VH platform including 80960JT core initialization, Memory Controller initialization, memory initialization, and PCI Address Translation Unit (ATU) initialization.

The EVAL80960VH platform is designed to use the Configuration Mode of the 80960VH processor. Configuration Mode allows the 80960JT core to initialize and control the initialization process before the PCI host configures the 80960VH processor. By utilizing Configuration Mode, the user is given the ability to initialize the PCI configuration registers to values other than the default power-up values. Configuration Mode gives the user maximum flexibility to customize the way in which the 80960VH processor and EVAL80960VH platform appear to the PCI host configuration software.

5.1.2 80960 JT Core Initialization

The 80960JT core begins the initialization process by reading its Initial Memory Image (IMI) from a fixed address in the boot ROM (FEFF FF30H in the i960 core address space). The IMI includes the Initialization Boot Record (IBR), the Process Control Block (PRCB), and several system data structures. The IBR provides initial configuration information for the core and integrated peripherals, pointers to the system data structures and the first instruction to be executed after processor initialization, and checksum words that the processor uses in its self-test routine. In addition to the IBR and PRCB, the required data structures are the:
5.1.3 Memory Controller Initialization

Since the i960® VH embedded-PCI processor Memory Controller is integral to the design and operation of the EVAL80960VH platform, the operational parameters for Bank 0 and Bank 1 are established immediately after processor core initialization. Memory Bank 0 is associated with the ROM on the EVAL80960VH platform. Memory Bank 1 is associated with the UART and the LED Control Register. Parameters such as Bank Base Address, Read Wait States, and Write Wait States must be established to ensure the proper operation of the EVAL80960VH platform. The Memory Controller is initialized so as to be consistent with the EVAL80960VH platform memory map shown in Figure 4-2.

5.1.4 PCI Interface Initialization

On the PCI bus, the ATU provides access capability between the PCI bus and the local i960 processor bus.

The platform can be initialized into one of three modes. Modes 0, 1, and 2 are described below.

Table 5-1. Initialization Modes

<table>
<thead>
<tr>
<th>RST_MODE#/S1-1</th>
<th>RETRY#/S1-1</th>
<th>Initialization Mode</th>
<th>PCI Interface</th>
<th>i960 Core Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/ON</td>
<td>0/ON</td>
<td>Mode 0</td>
<td>Accepts Transactions</td>
<td>Held in Reset</td>
</tr>
<tr>
<td>1/OFF</td>
<td>0/ON</td>
<td>Mode 1</td>
<td>Accepts Transactions</td>
<td>Initializes</td>
</tr>
<tr>
<td>1/OFF</td>
<td>1/OFF</td>
<td>Mode 2 (Default)</td>
<td>Retries All Configuration Transactions</td>
<td>Initializes</td>
</tr>
</tbody>
</table>

5.1.4.1 Mode 0

Mode 0 allows a host processor to configure the i960® VH embedded-PCI processor while the i960 core processor is held in rest.

The host processor can allocate PCI address space, and assign IRQ numbers. The Memory Controller and ATU can be initialized by the host processor. Program code for the i960 core processor may be downloaded into local memory by the host processor.

The host processor clears the i960 core processor reset by clearing the Core Processor Reset bit in the RRCR. This deasserts the reset signal on the i960 core processor and the processor begins its initialization process.
5.1.4.2 Mode 1

Mode 1 allows each unit of the i960® VH embedded-PCI processor to be initialized in its own manner. All units are reset when the \texttt{P_RST#} signal is asserted. Each unit returns to its default state. Be aware that race conditions may exist between i960 core processor operation after reset and PCI configuration.

5.1.4.3 Mode 2

This is the default initialization mode.

Mode 2 allows the i960 core processor to initialize and control the initialization process before the host processor is allowed to configure the 80960VH processor. The \textit{i960® VH Processor Developer’s Manual} describes the i960 core processor initialization process. This option is only available if the i960 core processor Initial Memory Image (IMI) is properly loaded into memory at the correct location.

The host processor is prevented from initializing the 80960VH processor. During this time, the ATU will signal a retry on all configuration cycles it receives on the primary PCI bus until the i960 core processor has cleared the Configuration Cycle Disable bit in the RRCR.

By allowing the i960 core processor to control the initialization process, it is possible to initialize the PCI configuration registers to values other than the default power-up values. Certain PCI configuration registers that are read only through PCI configuration cycles are read/write from the i960 core processor. This allows the programmer to customize the way the i960® VH embedded-PCI processor appears to the PCI configuration software.

5.1.5 ATU Initialization

The ATU initialization includes initialization by the i960 JT core and initialization by the PCI host processor. Local initialization occurs first and consists mainly of establishing the operational parameters for access to the EVAL80960VH platform local bus. The Primary Inbound ATU Limit Register (PIALR) is initialized to establish the block size of memory required by the ATU. The PIALR value is based on the installed DRAM configuration. The Primary Inbound ATU Translate Value Register (PIATVR) is initialized to establish the translation value to convert PCI addresses to local addresses. The Primary Outbound Memory Window Value Register (POMWVR) is initialized to establish the translation value for Local-to-PCI accesses. The POMWVR value remains at its default value of “0” to allow the EVAL80960VH platform to access the start of the PCI Memory address map, which is typically occupied by PCI host memory. Likewise, the Primary Outbound I/O Window Value Register (POIOWVR) remains at its default value of “0” to allow the EVAL80960VH platform to access the start of the PCI I/O address map. PCI Doorbell-related parameters are also established to allow for communication between the EVAL80960VH platform and a PCI bus master using the doorbell mechanism.

By default, Primary Outbound Configuration Cycle parameters are not established and Dual Address Cycle (DAC) support is not enabled. The ATU Configuration Register (ATUCR) is initialized to establish the operational parameters for the Doorbell Unit and ATU interrupts and to enable the ATU. The PCI host is responsible for allocating PCI address space (Memory, Memory Mapped I/O, and I/O), and assigning the PCI Base addresses for the EVAL80960VH platform.
5.2 mon960 Kernel

The mon960 Kernel (monitor) provides the EVAL80960VH user with a software platform on which application software can be developed and run. The monitor provides several features that the EVAL80960VH user can use to speed application development. Among the available features are:

- Communication with a terminal or terminal emulation package on a host computer through a serial cable with automatic baud rate detection
- Communication with a symbolic debugger such as gdb960 (available from Intel) using the Host Debugger Interface (HDI) software. gdb960 or mondb can communicate through serial, PCI or JTAG interfaces.
- Communication with the host computer via the PCI bus
- Downloads of object files via the PCI, serial or JTAG port
- Downloads of ELF object files via the PCI bus
- On-board erasure and programming of Intel 28F016S5 Flash ROM at U14
- Memory display and modification capability
- Breakpoint and single-step capability to support debugging of user code
- Disassembly of i960 processor instructions

5.3 Diagnostics

EVAL80960VH platform diagnostic routines serve a two-fold purpose: to verify proper hardware operation and to provide example code for users who need similar functions in their applications.

5.3.1 Board Level Diagnostics

Board level diagnostics exercise all basic areas of the EVAL80960VH platform. Diagnostic routines include DRAM tests, UART tests, LED tests, internal timer tests, PCI bus tests, and PCI bus tests. PCI bus tests exercise the ATU, the PCI Doorbell unit, and the PCI DMA controller. Interrupts from both local and PCI sources are generated and handled. The PCI bus tests require an external test suite running on a PC to verify complete functionality of the EVAL80960VH platform.

The tests can be invoked using the ‘po’ command in either Terminal Emulation Mode or from mondb. This command is explained in Chapter 4, Monitor Commands of the MON960 Debug Monitor User’s Guide.
This appendix identifies all components on the EVAL80960VH Evaluation Platform (Table A-1).

<table>
<thead>
<tr>
<th>Location</th>
<th>Cyclone P/N</th>
<th>Part Description</th>
<th>Qty</th>
<th>Mfg</th>
<th>Mfg Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>U7</td>
<td>100-1198</td>
<td>IC/SM 74ALS08 (SOIC 14)</td>
<td>1</td>
<td>National Semiconductor</td>
<td>DM74ALS08M</td>
</tr>
<tr>
<td>U3</td>
<td>100-1199</td>
<td>IC/SM 74ALS04 SOIC</td>
<td>1</td>
<td>National Semiconductor</td>
<td>DM74ALS04BM</td>
</tr>
<tr>
<td>U2</td>
<td>100-1792</td>
<td>IC/SM 74ABT273 SOIC</td>
<td>1</td>
<td>Texas Instruments</td>
<td>SN74ABT273DW</td>
</tr>
<tr>
<td>U15</td>
<td>100-1793</td>
<td>IC/SM 74ABT573 SOIC</td>
<td>1</td>
<td>Texas Instruments</td>
<td>SN74ABT573DW</td>
</tr>
<tr>
<td>U4</td>
<td>100-2048</td>
<td>IC / SM 1488A SOIC</td>
<td>1</td>
<td>National Semiconductor</td>
<td>DS1488M</td>
</tr>
<tr>
<td>U1</td>
<td>100-2049</td>
<td>IC / SM 1489A SOIC</td>
<td>1</td>
<td>National Semiconductor</td>
<td>DS1489AM</td>
</tr>
<tr>
<td>U10</td>
<td>100-5098</td>
<td>IC/SM MAX767CAP SOIC</td>
<td>1</td>
<td>Maxim</td>
<td>MAX767CAP</td>
</tr>
<tr>
<td>U12</td>
<td>102-1055-99</td>
<td>Processor 80960VH from INTEL</td>
<td>1</td>
<td>Intel</td>
<td>102-1055-99</td>
</tr>
<tr>
<td>U6</td>
<td>102-1127</td>
<td>VLSI I/O UART 16C550 PLCC</td>
<td>1</td>
<td>Texas Instruments</td>
<td>TL16C550AFN</td>
</tr>
<tr>
<td>U14</td>
<td>103-1265</td>
<td>MEM Flash E28F016S5-090 TSOP</td>
<td>1</td>
<td>Intel</td>
<td>E28F016S5-090</td>
</tr>
<tr>
<td>U8</td>
<td>103-1550</td>
<td>MEMORY EEPROM 24C08 SOIC 8</td>
<td>1</td>
<td>Microchip</td>
<td>24LC08SOI</td>
</tr>
<tr>
<td>C1-C10,C11, C13,C16-C23, C25,C26,C30, C32,C33, C36-C38, C43-C52</td>
<td>110-3301</td>
<td>CAP CERM SM, 0.01uf (0805)</td>
<td>38</td>
<td>Kemet</td>
<td>C0805C103K5RAC</td>
</tr>
<tr>
<td>C24,C31,C41, C42</td>
<td>110-3304</td>
<td>CAP CERM SM, 0.1uf (0805)</td>
<td>4</td>
<td>Kemet</td>
<td>C0805C104K5RAC</td>
</tr>
<tr>
<td>R15,R18,R30</td>
<td>126-1001-05</td>
<td>R/SM 1/10W 5% 10 ohm (0805)</td>
<td>3</td>
<td>Dale</td>
<td>CRCW080510R0F</td>
</tr>
<tr>
<td>R1,R2</td>
<td>126-1003-05</td>
<td>R/SM 1/10W 5% 1K ohm (0805)</td>
<td>2</td>
<td>Dale</td>
<td>CRCW08051001FRT</td>
</tr>
<tr>
<td>R12,R19,R20, R23,R25,R29</td>
<td>126-1004-05</td>
<td>R/SM 1/10W 5% 10K ohm (0805)</td>
<td>6</td>
<td>Dale</td>
<td>CRCW08051002FRT</td>
</tr>
<tr>
<td>R4,R5</td>
<td>126-4702-05</td>
<td>R/SM 1/10W 5% 470 ohm (0805)</td>
<td>2</td>
<td>Dale</td>
<td>CRCW 0805 471JT</td>
</tr>
</tbody>
</table>
### Table A-1. EVAL80960VH Bill of Materials (Sheet 2 of 3)

<table>
<thead>
<tr>
<th>Location</th>
<th>Cyclone P/N</th>
<th>Part Description</th>
<th>Qty</th>
<th>Mfg</th>
<th>Mfg Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R24</td>
<td>127-0000-05</td>
<td>R/SM 1/8W 5% 000 ohm chip 1206</td>
<td>1</td>
<td>Dale</td>
<td>CRCW1206000Z</td>
</tr>
<tr>
<td>J4-J6</td>
<td>130-1438</td>
<td>CONN SM/TH Mictor 38P Receptacle</td>
<td>3</td>
<td>Amp</td>
<td>767054-1</td>
</tr>
<tr>
<td>J3</td>
<td>130-1450</td>
<td>CONN SIMM 72P Socket / Angled</td>
<td>1</td>
<td>Amp</td>
<td>822134-3</td>
</tr>
<tr>
<td>J1</td>
<td>130-1509</td>
<td>CONN TJ6 PCB 6/6 LP Thru hole</td>
<td>1</td>
<td>Kycon</td>
<td>GM-N-66</td>
</tr>
<tr>
<td>J2</td>
<td>130-1577</td>
<td>CONN Hdr 16 pin/w shell, pcb</td>
<td>1</td>
<td>Amp</td>
<td>103308-3</td>
</tr>
<tr>
<td>Z1</td>
<td>130-1621</td>
<td>Jumper JUMP2X1</td>
<td>1</td>
<td>Molex</td>
<td>22-28-4023</td>
</tr>
<tr>
<td>J7,J8</td>
<td>130-1717</td>
<td>CONN 2x25 Socket Strip .025&quot;</td>
<td>2</td>
<td>Samtec</td>
<td>BHT-125-04-GF-D</td>
</tr>
<tr>
<td>SW1</td>
<td>150-4000</td>
<td>Switch, SW-PUSH 3P</td>
<td>1</td>
<td>Epson</td>
<td>EP12SD1ABE</td>
</tr>
<tr>
<td>U5</td>
<td>150-6009</td>
<td>OSC 1.8432MHz 1/2 - Thru hole</td>
<td>1</td>
<td>Kyocera</td>
<td>KH0HC1CSE 1.843</td>
</tr>
<tr>
<td>U11</td>
<td>150-8005</td>
<td>Clock Chip CY7B9910-7SC</td>
<td>1</td>
<td>Cypress</td>
<td>CY7B9910-7SC</td>
</tr>
<tr>
<td>CR1</td>
<td>150-9100</td>
<td>LED Green</td>
<td>1</td>
<td>Hewlett Packard</td>
<td>HLMP-3507$010</td>
</tr>
<tr>
<td>CR2</td>
<td>150-9101</td>
<td>LED-Red</td>
<td>1</td>
<td>Hewlett Packard</td>
<td>HLMP3301$010 HP</td>
</tr>
<tr>
<td>CR3,CR4</td>
<td>150-9103</td>
<td>LED-Red-Small Group</td>
<td>2</td>
<td>Dialight</td>
<td>555-4001</td>
</tr>
<tr>
<td>U9</td>
<td>193-2001</td>
<td>SOCKET PLCC32 LP Surface Mount</td>
<td>1</td>
<td>Amp</td>
<td>822273-1</td>
</tr>
<tr>
<td>U13</td>
<td>193-2002</td>
<td>SOCKET PLCC44 LP Surface Mount</td>
<td>1</td>
<td>Amp</td>
<td>822275-1</td>
</tr>
<tr>
<td>270-0938</td>
<td>PCB 0938A for VH Eval Board</td>
<td>1</td>
<td>N/A</td>
<td>270-0938</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>101-0921</td>
<td>PAL MACH111-15JC / PLCC44</td>
<td>1</td>
<td>Vantis</td>
<td>MACH111-15JC</td>
</tr>
<tr>
<td>N/A</td>
<td>103-1214-99</td>
<td>MEMORY N28F020-150 / Intel</td>
<td>1</td>
<td>Intel</td>
<td>N28F020-150</td>
</tr>
<tr>
<td>N/A</td>
<td>230-1102</td>
<td>SCREW 4.40x3/16 Pan Head Slot</td>
<td>2</td>
<td>McMaster Carr</td>
<td>91792A105</td>
</tr>
<tr>
<td>N/A</td>
<td>230-1199</td>
<td>SCREW NYLON 4/ 40x3/16 Pan Head</td>
<td>2</td>
<td>McMaster Carr</td>
<td>94611A105 (100)</td>
</tr>
<tr>
<td>N/A</td>
<td>230-1503</td>
<td>HARDWARE 4-40 x 0.465 Spacer</td>
<td>2</td>
<td>Keystone</td>
<td>1902A at 0.465</td>
</tr>
<tr>
<td>N/A</td>
<td>351-0938</td>
<td>Faceplate PCI938</td>
<td>1</td>
<td>Gompf</td>
<td>9334-0406</td>
</tr>
<tr>
<td>C29</td>
<td>110-3103</td>
<td>CAP SM, 0.22uf (1206)</td>
<td>1</td>
<td>Philips</td>
<td>12062E224M9BB2</td>
</tr>
<tr>
<td>C15</td>
<td>110-3208</td>
<td>CAP TANT SM 220uf, 10V (7343)</td>
<td>1</td>
<td>AVX</td>
<td>TPSE227K010R010</td>
</tr>
<tr>
<td>C28,C35,C39, C40</td>
<td>110-3209</td>
<td>CAP TANT SM 47uf, 16V (7343)</td>
<td>4</td>
<td>AVX</td>
<td>TPS476K016R015</td>
</tr>
</tbody>
</table>
Table A-1. EVAL80960VH Bill of Materials (Sheet 3 of 3)

<table>
<thead>
<tr>
<th>Location</th>
<th>Cyclone P/N</th>
<th>Part Description</th>
<th>Qty</th>
<th>Mfg</th>
<th>Mfg Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>C12,C14,C27, C34</td>
<td>110-3211</td>
<td>CAP TANT SM 4.7uf, 35V (7343)</td>
<td>4</td>
<td>AVX</td>
<td>293D475X9035D2T</td>
</tr>
<tr>
<td>R28</td>
<td>128-0110</td>
<td>Resistor/SM 1 W 5% 0.04 ohm</td>
<td>1</td>
<td>Dale</td>
<td>WSL-2512R04-1TR</td>
</tr>
<tr>
<td>R26</td>
<td>128-0120</td>
<td>Resistor/SM 1/2W 5% 100 ohm</td>
<td>1</td>
<td>Beckman</td>
<td>BCR 1/2 101 JT</td>
</tr>
<tr>
<td>R13</td>
<td>129-2101</td>
<td>Resistor Pk SM RNC4R8P 2.7kohm</td>
<td>1</td>
<td>CTS</td>
<td>742083272JTR</td>
</tr>
<tr>
<td>R3,R16,R17, R21</td>
<td>129-2102</td>
<td>Resistor Pk SM RNC4R8P 10kohm</td>
<td>4</td>
<td>Philips</td>
<td>2350-034-10103</td>
</tr>
<tr>
<td>R8-R11,R22, R27,R31</td>
<td>129-2103</td>
<td>Resistor Pk SM RNC4R8P 22ohm</td>
<td>7</td>
<td>CTS</td>
<td>742083220JTR</td>
</tr>
<tr>
<td>R6,R7</td>
<td>129-2104</td>
<td>Resistor Pk SM RNC4R8P 470ohm</td>
<td>2</td>
<td>CTS</td>
<td>742083471JTR</td>
</tr>
<tr>
<td>R14</td>
<td>129-2105</td>
<td>Resistor Pk SM RNC4R8P 4.7kohm</td>
<td>1</td>
<td>CTS</td>
<td>742083472JTR</td>
</tr>
<tr>
<td>L1</td>
<td>150-1010</td>
<td>Coil 10 uh SM CDR74B-100MC</td>
<td>1</td>
<td>Sumida</td>
<td>CDR74B-100MC</td>
</tr>
<tr>
<td>S1,S2</td>
<td>150-4050</td>
<td>Switch/SM DIP4 Mors# DHS-4S</td>
<td>2</td>
<td>Morse</td>
<td>DHS-4S</td>
</tr>
<tr>
<td>CR5</td>
<td>160-1101</td>
<td>Diode 1N5817 Surface Mount</td>
<td>1</td>
<td>Central Semiconductor</td>
<td>CMSH1-20</td>
</tr>
<tr>
<td>CR6</td>
<td>160-1102</td>
<td>Diode CMPSH3 Surface Mount</td>
<td>1</td>
<td>Central Semiconductor</td>
<td>CMPSH3</td>
</tr>
<tr>
<td>Q1</td>
<td>160-1103</td>
<td>Diode IRF7101 Surface Mount</td>
<td>1</td>
<td>International Rectifier</td>
<td>IRF7101T1</td>
</tr>
</tbody>
</table>
MODULE VH
TITLE 'VH'
" PATTERN 101-XXXX-01
" REVISION
" AUTHOR John Neumann
" COMPANY Cyclone Microsystems
" DATE June 10, 1998
" CHIP MACH111-15JC

CLK PIN 11;  "pal clock input
CE1b PIN 2;  "ROM space chip enable #1
RSTb PIN 3;  "local reset
WRDb PIN 4;  "write/read strobe
AD0 PIN 5;  "address/data bit 0
AD1 PIN 6;  "address/data bit 1
CE0b PIN 7;  "ROM space chip enable #0
ROMSWAP PIN 8;  "Select between EEPROM devices
ROMDIS PIN 14;  "EEPROM ROM disable
RA18 PIN 9;  "ROM address bit 18
RA21 PIN 10;  "ROM address bit 21
RESET PIN 17;  "Push Button reset
SET PIN 18;   "Push Button set
PBRSTb PIN 29;  "Debounce push button reset
QS0 PIN 43 ISTYPE 'REG_D'; "quickswitch mux select 0
QS1 PIN 42 ISTYPE 'REG_D'; "quickswitch mux select 1
IORb PIN 41 ISTYPE 'REG_D'; "UART read strobe
IOWb PIN 40 ISTYPE 'REG_D'; "UART write strobe
SELLEDb PIN 39 ISTYPE 'REG_D'; "LED chip enable
SELUARTb PIN 38 ISTYPE 'REG_D'; "UART chip enable
SELROMAb PIN 37 ISTYPE 'REG_D'; "Select for 28F020
SELROMBb PIN 36 ISTYPE 'REG_D'; "Select for 28F016S5

EQUATIONS
" Quickswitch mux selects for DRAM select, write to address E0200000h with data
QS0 := WRDb & RSTb & RA21 * !CE1b & AD0
# QS0 & !(RA21 & !CE1b) & RSTb*hold
# QS0 & !WRDb & RSTb & RA21 & !CE1b;hold during a read
QS0.CLK = CLK;
QS1 := WRDb & RSTb & RA21 * !CE1b & AD1
# QS1 & !(RA21 & !CE1b) & RSTb*hold
# QS1 & !WRDb & RSTb & RA21 & !CE1b;hold during a read
QS1.CLK = CLK;

" Select LED is write only to address E0040000h
!SELLEDb := RA18 & !RA21 & WRDb & !CE1b;
SELLEDb.CLK = CLK;

" The UART is located at address E0000000h
!SELUARTb := !RA18 & !RA21 & !CE1b;
SELUARTb.CLK = CLK;

" UART read and write strobes
!IORb := !WRDb & RSTb & !CE1b & !SELUARTb;
IORb.CLK = CLK;
!IOWb := WRDb & RSTb & !CE1b & !SELUARTb;
IOWb.CLK = CLK;
* Flash ROM selects. Rom A is located at FEC0 0000H to FEC3 FFFFH.
* ROM B is located at FEE0 0000H to FEFF FFFFH.
* The equations below allow ROM 'B' and ROM 'A' to be 'swapped' thus
  changing the device from which the i960 boots.
!SELROMAb := ROMSWAP & !CE0b & !RA21 & ROMDIS
  # !ROMSWAP & !CE0b & RA21 & ROMDIS;
SELROMAb.CLK = CLK;
!SELROMBb := ROMSWAP & !CE0b & RA21 & ROMDIS
  # !ROMSWAP & !CE0b & RA21 & ROMDIS;
SELROMBb.CLK = CLK;
* Push button reset must be debounce before it can be anded with PCI reset
!PBRSTb = !SET # (!PBRSTb & RESET);
END
D.1 Introduction

The Backplane 0015 board from Cyclone Microsystems allows the EVAL80960VH platform to be installed into a PCI socket on the board and function as a host system board. The Backplane 0015 contains four PCI slots, J1 through J4. Figure D-1 depicts board component locations. The PCI backplane board contains a 33MHz bus clock, arbitration for the four PCI slots, an ATX power supply connector with stand-by voltage power on circuit, a reset circuit, and two LEDs indicating +5V and +3.3V availability.

D.2 Installing the EVAL80960VH into the Backplane 0015

The EVAL80960VH platform must be installed into slot J4 of the backplane board. The enclosed ribbon cable must be connected to allow proper function of the EVAL80960VH as a host board. Please refer to Figure D-2 and Figure D-3 during installation and setup. Install the 26 pin keyed connector into the header position, J5, on the backplane board. The red wire should line up with pin 1 of J5. Next install the other end of the ribbon cable into J7 on the EVAL80960VH. Take care as this connector does not have any keying associated with it. The red wire should line up with pin 50 on J7, towards the top edge of the platform. This cable allows interrupts from the PCI slots on the backplane board to reach the i960VH processor.

D.3 Powering the EVAL80960VH/Backplane System

The PCI backplane board is designed to be powered via an ATX power supply. Before inserting the power cable into the ATX power connector on the backplane board make sure that the +5VSB (stand-by voltage) is off. Inserting the power cable with +5VSB on may cause damage to the start up circuit and the backplane board may not power up. After the connector is installed, depress the momentary power switch to turn on the +5V and +3.3V rails on the ATX power supply.
D.4 Interrupt Routing and IDSELs on the Backplane 0015

The PCI slots J1 through J3 allow up to 3 adapter boards to be installed in the system. These adapter boards are configured by the EVAL80960VH and interrupt the i960VH processor through the ribbon cable. The interrupt routing and IDSELs on the backplane board is shown in Table D-1. Note that the EVAL80960VH has the ability to interrupt itself.

Table D-1. PCI Interrupt Routing and IDSELs

<table>
<thead>
<tr>
<th>SLOT</th>
<th>INTA</th>
<th>INTB</th>
<th>INTC</th>
<th>INTD</th>
<th>IDSEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>XINT1</td>
<td>XINT2</td>
<td>XINT3</td>
<td>XINT0</td>
<td>AD16</td>
</tr>
<tr>
<td>J2</td>
<td>XINT2</td>
<td>XINT3</td>
<td>XINT0</td>
<td>XINT1</td>
<td>AD17</td>
</tr>
<tr>
<td>J3</td>
<td>XINT3</td>
<td>XINT0</td>
<td>XINT1</td>
<td>XINT2</td>
<td>AD18</td>
</tr>
<tr>
<td>J4</td>
<td>XINT0</td>
<td>XINT1</td>
<td>XINT2</td>
<td>XINT3</td>
<td>AD19</td>
</tr>
</tbody>
</table>

Figure D-1. EVAL80960VH Backplane Board Illustration
Figure D-2. EVAL80960VH/Backplane 0015 with Eval Board Installed

Figure D-3. EVAL80960VH/Backplane 0015 Installation
D.5 PCI Host Detection and Configuration

When installed in the slot J4 of the backplane, the EVAL80960VH acts as the PCI host in the system. The EVAL80960VH determines whether it is in the host state by the XINT4 input. The backplane drives the VH processor’s XINT4 low via the ribbon cable. Upon system startup, MON960 uses the presence of an XINT4 interrupt to determine whether it needs to initialize the primary PCI bus. If the EVAL80960VH is the PCI host, the ATU base address (PIABAR) is set to the base of DRAM (A000 0000h), and the master enable bit in the ATU command register (PATUCMD) is set. MON960 then disables XINT4. Users should be aware that XINT4 should not be enabled.

D.6 PCI Initialization

When acting as a PCI host, MON960 extensions are responsible for initializing the devices on the PCI bus of the EVAL80960VH. PCI initialization involves allocating address spaces (Memory, Memory Mapped I/O, and I/O), assigning PCI base addresses, assigning IRQ values (see Table D-1), and enabling PCI mastership. Devices containing PCI-to-PCI bridges and hierarchical buses are not supported.

D.7 PCI BIOS Routines

The PCI BIOS routines are accessible at the MON960 layer and can also be accessed by application level software by using the i960 system call mechanism and the System Procedure Table. The supported BIOS functions are described in the subsections that follow.

```c
pci_bios_present()
find_pci_device()
find_pci_class_code()
generate_special_cycle()
read_config_byte()
read_config_word()
read_config_dword()
write_config_byte()
write_config_word()
write_config_dword()
get_irq_routing_options()
set_pci_irq()
```

Although the calling interface is different from that used on a DOS-based host, these functions preserve, as closely as possible, the parameters and return values described in *PCI Local Bus Specification*, Revision 2.1. Functions that return multiple values do so by filling in the fields of a structure passed by the calling routine.

```c
pci_bios_present
```
This function allows the caller to determine whether the PCI BIOS interface function set is present, and the current interface version level. It also provides information about the hardware mechanism used for accessing configuration space and whether or not the hardware supports generation of PCI Special Cycles.

Calling convention:

```c
int pci_bios_present ( 
    PCI_BIOS_INFO *info 
);
```

Return values:

This function always returns SUCCESSFUL.

### find_pci_device

This function returns the location of PCI devices that have a specific Device ID and Vendor ID. Given a Vendor ID, a Device ID, and an Index, the function returns the Bus Number, Device Number, and Function Number of the Nth Device/Function whose Vendor ID and Device ID match the input parameters.

Calling software can find all devices having the same Vendor ID and Device ID by making successive calls to this function starting with the index set to “0”, and incrementing the index until the function returns DEVICE_NOT_FOUND. A return value of BAD_VENDOR_ID indicates that the Vendor ID value passed had a value of all “1”s.

Calling convention:

```c
int find_pci_device ( 
    int     device_id, 
    int     vendor_id, 
    int     index 
);
```

Return values:

This function returns SUCCESSFUL if the indicated device is located, DEVICE_NOT_FOUND if the indicated device cannot be located, or BAD_VENDOR_ID if the vendor_id value is illegal.

### find_pci_class_code

This function returns the location of PCI devices that have a specific Class Code. Given a Class Code and an Index, the function returns the Bus Number, Device Number, and Function Number of the Nth Device/Function whose Class Code matches the input parameters.

Calling software can find all devices having the same Class Code by making successive calls to this function starting with the index set to “0”, and incrementing the index until the function returns DEVICE_NOT_FOUND.

Calling convention:

```c
int find_pci_class_code ( 
    int     class_code, 
    int     index 
);
```
Return values:

This function returns SUCCESSFUL if the indicated device is located or DEVICE_NOT_FOUND if the indicated device cannot be located.

**generate_special_cycle**

This function allows for generation of PCI Special Cycles. The generated special cycle is broadcast on a specific PCI Bus in the system.

PCI Special Cycles are not supported on the 0015 backplane PCI bus.

Calling convention:

```c
int generate_special_cycle (
    int bus_number,
    int special_cycle_data
);
```

Return values:

Since PCI Special Cycles are not supported by the 0015 backplane, this function always returns FUNC_NOT_SUPPORTED.

**read_config_byte**

This function allows the caller to read individual bytes from the configuration space of a specific device.

Calling convention:

```c
int read_config_byte (
    int bus_number,
    int device_number,
    int function_number,
    int register_number,/* 0,1,2,...,255 */
    UINT 8*data
);
```

Return values:

This function returns SUCCESSFUL if the indicated byte was read correctly or ERROR if there is a problem with the parameters.

**read_config_word**

This function allows the caller to read individual shorts (16 bits) from the configuration space of a specific device. The Register Number parameter must be a multiple of two (i.e., bit 0 must be set to "0").

Calling convention:

```c
int read_config_word (
    int bus_number,
    int device_number,
    int function_number,
    int register_number,/* 0,2,4,...,254 */
```
UINT 16*data 
);

Return values:
This function returns SUCCESSFUL if the indicated word was read correctly or ERROR if there is a problem with the parameters.

**read_config_dword**

This function allows the caller to read individual longs (32 bits) from the configuration space of a specific device. The Register Number parameter must be a multiple of four (i.e., bits 0 and 1 must be set to “0”).

Calling convention:
int read_config_dword (  
    int bus_number,  
    int device_number,  
    int function_number,  
    int register_number,/* 0,4,8,...,252 */  
    UINT 32*data  
);

Return values:
This function returns SUCCESSFUL if the indicated long was read correctly or ERROR if there is a problem with the parameters.

**write_config_byte**

This function allows the caller to write individual bytes to the configuration space of a specific device.

Calling convention:
int write_config_byte (  
    int bus_number,  
    int device_number,  
    int function_number,  
    int register_number,/* 0,1,2,...,255 */  
    UINT 8*data  
);

Return values:
This function returns SUCCESSFUL if the indicated byte was written correctly or ERROR if there is a problem with the parameters.

**write_config_word**

This function allows the caller to write individual shorts (16 bits) to the configuration space of a specific device. The Register Number parameter must be a multiple of two (i.e., bit 0 must be set to “0”).

Calling convention:
int write_config_word {
    int bus_number,
    int device_number,
    int function_number,
    int register_number, /* 0, 2, 4, ..., 254 */
    UINT 16*data
};

Return values:

This function returns SUCCESSFUL if the indicated word was written correctly or ERROR if there is a problem with the parameters.

**write_config_dword**

This function allows the caller to write individual longs (32 bits) to the configuration space of a specific device. The Register Number parameter must be a multiple of four (i.e., bits 0 and 1 must be set to “0”).

Calling convention:

int write_config_dword {
    int bus_number,
    int device_number,
    int function_number,
    int register_number, /* 0, 4, 8, ..., 252 */
    UINT 32*data
};

Return values:

This function returns SUCCESSFUL if the indicated long was written correctly or ERROR if there is a problem with the parameters.

**get_irq_routing_options**

This routine returns the PCI interrupt routing options available on the 0015 backplane. Two values are provided for each PCI interrupt pin for each device. One of these values is a bitmap that shows the interrupt to which 80960VH XINT (XINT3:0) is connected. The second value is a link value that provides a way of specifying which PCI interrupt pins are wire-OR’ed together on the motherboard. Interrupt pins with the same link value are wired together.

The PCI Interrupt routing fabric on the 0015 backplane is not reconfigurable (fixed mapping relationships).

Calling convention:

int get_irq_routing_options {
    PCI_IRQ_ROUTING_TABLE *table
};

Return values:

This function always returns SUCCESSFUL.

**set_pci_irq**
The PCI Interrupt routing fabric on the 0015 backplane is not reconfigurable (fixed mapping relationships); therefore, this function is not supported.

Calling convention:

```c
int set_pci_irq (
    int int_pin,
    int irq_num,
    int bus_dev
);
```

Return values:

This function always returns FUNC_NOT_SUPPORTED.

### D.8 Additional MON960 Commands

The following commands have been added to the UI interface of MON960 to support the 0015 backplane.

**print_pci Utility**

A `print_pci` command to MON960 is accessed through the MON960 command prompt. This command displays the contents of the PCI configuration space on a selected adapter on the primary PCI interface or on the i960 VH processor itself. For more information on the meaning of the fields in PCI configuration space, refer to the *PCI Local Bus Specification* Revision 2.1. The syntax of this command is:

`pp <bus number> <device number> <function number>`

### D.9 Bill of Material

This appendix identifies all components on the Backplane 0015 Interface (Table D-2).

<table>
<thead>
<tr>
<th>Cyclone P/N</th>
<th>Reference Designator</th>
<th>Description</th>
<th>Qty</th>
<th>Mfg</th>
<th>Mfg Part Number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-1698</td>
<td>U7</td>
<td>IC/SM 74HC74 SOIC-14</td>
<td>1</td>
<td>Texas Instruments</td>
<td>SN74HC74D</td>
<td></td>
</tr>
<tr>
<td>100-1699</td>
<td>U5</td>
<td>IC/SM 74HC14 SOIC</td>
<td>1</td>
<td>National Semi.</td>
<td>74HC14</td>
<td></td>
</tr>
<tr>
<td>100-1225</td>
<td>U4</td>
<td>IC/SM 74AS760 SOIC</td>
<td>1</td>
<td>Texas Instruments</td>
<td>SN74AS760DW</td>
<td></td>
</tr>
<tr>
<td>100-2200</td>
<td>U6</td>
<td>IC/SM TL7705ACD</td>
<td>1</td>
<td>Texas Instruments</td>
<td>TL7705ACD</td>
<td></td>
</tr>
<tr>
<td>100-3098</td>
<td>U3</td>
<td>IC/SM LVCMOS Fanout Buffr SSOP</td>
<td>1</td>
<td>Motorola</td>
<td>MPC9140</td>
<td></td>
</tr>
<tr>
<td>110-3209</td>
<td>C16,C31</td>
<td>CAP TANT SM 47uf, 16V (7343)</td>
<td>2</td>
<td>AVX</td>
<td>TPSD476K016R015</td>
<td>(2)</td>
</tr>
</tbody>
</table>
Table D-2. Backplane 0015 Interface Bill of Material (Sheet 2 of 3)

<table>
<thead>
<tr>
<th>Cyclone P/N</th>
<th>Reference Designator</th>
<th>Description</th>
<th>Qty</th>
<th>Mfg</th>
<th>Mfg Part Number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>110-3210</td>
<td>C38</td>
<td>CAP TANT SM 33uf, 10V (7343)</td>
<td>1</td>
<td>Sprague</td>
<td>293D336X9016D2T</td>
<td>(2)</td>
</tr>
<tr>
<td>110-3214</td>
<td>C35</td>
<td>CAP TANT SM 10uf 25/35V (7343)</td>
<td>1</td>
<td>Sprague</td>
<td>293D1060025D2T</td>
<td>(2)</td>
</tr>
<tr>
<td>110-3301</td>
<td>C8-C15,C18-C25</td>
<td>CAP CERM SM, 0.01uf (0805)</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110-3304</td>
<td>C1-C7,C17,C26-C30,C32-C34,C36,C37</td>
<td>CAP CERM SM, 0.1uf (0805)</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>126-0000-05</td>
<td>R21</td>
<td>R/SM 1/10W 5% 000 ohm (0805)</td>
<td>1</td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>126-1004-05</td>
<td>R5,R13-R20,R22,R24</td>
<td>R/SM 1/10W 5% 10K ohm (0805)</td>
<td>11</td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>126-2703-05</td>
<td>R6,R23,R25</td>
<td>R/SM 1/10W 5% 2.7k ohm (0805)</td>
<td>3</td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>126-3301-05</td>
<td>R1,R9</td>
<td>R/SM 1/10W 5% 33 ohm (0805)</td>
<td>2</td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>126-3302-05</td>
<td>R4</td>
<td>R/SM 1/10W 5% 330 ohm (0805)</td>
<td>1</td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>126-4702-05</td>
<td>R3</td>
<td>R/SM 1/10W 5% 470 ohm (0805)</td>
<td>1</td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>129-2101</td>
<td>R2,R10-R12</td>
<td>Resistor Pk SM RNC4R8P 2.7kohm</td>
<td>4</td>
<td>CTS</td>
<td>742083272JTR</td>
<td>(2)</td>
</tr>
<tr>
<td>129-2105</td>
<td>R7</td>
<td>Resistor Pk SM RNC4R8P 4.7kohm</td>
<td>1</td>
<td>CTS</td>
<td>742083472JTR</td>
<td>(2)</td>
</tr>
<tr>
<td>129-2113</td>
<td>R8</td>
<td>Resistor Pk SM RNC4R8P 33 ohm</td>
<td>1</td>
<td>CTS</td>
<td>742083330JTR</td>
<td>(2)</td>
</tr>
<tr>
<td>129-2102</td>
<td>R301</td>
<td>Resistor Pk SM 10K ohm</td>
<td>1</td>
<td>CTS</td>
<td>742083103JTR</td>
<td></td>
</tr>
<tr>
<td>130-1460</td>
<td>J1-J4</td>
<td>CONN PCI SLOT 5V/PCB thru hole</td>
<td>4</td>
<td>AMP</td>
<td>145154-4</td>
<td></td>
</tr>
<tr>
<td>130-1519</td>
<td>J6</td>
<td>CONN PCPWR Male /Mini-Fit/20p</td>
<td>1</td>
<td>Molex</td>
<td>39-29-9202</td>
<td></td>
</tr>
<tr>
<td>130-1580</td>
<td>J5</td>
<td>CONN Hdr 26 pin/w shell, PCB</td>
<td>1</td>
<td>AMP</td>
<td>104339-6</td>
<td></td>
</tr>
<tr>
<td>130-1621</td>
<td>Z1,Z2</td>
<td>Jumper JUMP2X1</td>
<td>2</td>
<td>Molex</td>
<td>22-28-4023</td>
<td></td>
</tr>
<tr>
<td>150-4000</td>
<td>SW1,SW2</td>
<td>Switch, SW-PUSH 3P (Right Ang)</td>
<td>2</td>
<td>C&amp;K Switch</td>
<td>EP12SD1ABE</td>
<td></td>
</tr>
<tr>
<td>150-5013</td>
<td>U1</td>
<td>OSC 33.0 MHz Half</td>
<td>1</td>
<td>Ecliptek</td>
<td>EC1900HS3.000 M</td>
<td></td>
</tr>
<tr>
<td>150-9100</td>
<td>CR1,CR2</td>
<td>LED Green</td>
<td>2</td>
<td>Hewlett Packard</td>
<td>HLMP-3507$010</td>
<td></td>
</tr>
<tr>
<td>160-1108</td>
<td>CR3</td>
<td>Diode SM BAS16</td>
<td>1</td>
<td>National Semi.</td>
<td>BAS16</td>
<td>(2)</td>
</tr>
<tr>
<td>193-2002</td>
<td>U2</td>
<td>SOCKET PLCC44 LP Surface Mount</td>
<td>1</td>
<td>AMP</td>
<td>822275-1</td>
<td></td>
</tr>
<tr>
<td>230-1197</td>
<td>MT1-MT4</td>
<td>SCREW-NYLON 4/40x3/8&quot; Pan Slot</td>
<td>4</td>
<td>McMaster Carr</td>
<td>95000A108</td>
<td>(3)</td>
</tr>
<tr>
<td>230-1507</td>
<td>MT1-MT4</td>
<td>HARDWARE 4-40 x 3/8&quot; Standoff</td>
<td>4</td>
<td>Keystone</td>
<td>1902B (0.375&quot;)</td>
<td>(3)</td>
</tr>
<tr>
<td>270-0015</td>
<td></td>
<td>PCB 0015A / VH Extender</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: (1) - Provided by assembly house by specified description
Table D-2. Backplane 0015 Interface Bill of Material (Sheet 3 of 3)

<table>
<thead>
<tr>
<th>Cyclone P/N</th>
<th>Reference Designator</th>
<th>Description</th>
<th>Qty</th>
<th>Mfg</th>
<th>Mfg Part Number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note (2) - Provided by Cyclone - stocked at assembly house
Note (3) - Reference assembly drawing
Note (4) - Pre-programmed device

D.10 Schematics
D.11 PAL Code

TITLE PCI BUS ARBITRATION
PATTERN D703
REVISION C
AUTHOR Joe Niedermeyer
COMPANY Cyclone Microsystems
DATE 10-26-94

; D703B4 3-07-95 jpn Fixing PCI bus St Mach for retry.
; D703B 3-07-95 jpn Same as D703B4
; D703C1 3-08-95 jpn Adding TRDY*, LOCK* and STOP*, just in case.

CHIP D703 MACH210

;** INPUT PINS **;
pin 35 CLK ; 33MHz system clk

pin 10 HREQn ; Host (PCI9060) bus request
pin 11 AREQn ; PCI expansion slot "A" bus request
pin 13 BREQn ; PCI expansion slot "B" bus request
pin 32 CREQn ; PCI expansion slot "C" bus request
pin 33 DREQn ; PCI expansion slot "D" bus request

pin 41 FRAMEn ; PCI bus FRAME# signal
pin 3 TRDYn ; PCI bus target ready signal
pin 43 IRDYn ; PCI bus initiator ready signal
pin 2 STOPn ; PCI bus STOP# signal
pin 4 LOCKn ; PCI bus LOCK# signal
pin 42 ASYNCRSTn ; system reset, asynchronous, active low

;** OUTPUT PINS **;
pin 20 HGNTn REG ; Host (PCI9060) bus grant
pin 19 AGNTn REG ; PCI expansion slot "A" bus grant
pin 18 BGNTn REG ; PCI expansion slot "B" bus grant
pin 17 CGNTn REG ; PCI expansion slot "C" bus grant
pin 21 DGNTn REG ; PCI expansion slot "D" bus grant

pin 39 DUMMY1 COMB ; dummy output to use TRDY, LOCK and STOP

;** BURIED SIGNALS **;
NODE 3 P0 REG ; priority state variable
NODE 7 P1 REG ; priority state variable
NODE 11 P2 REG ; priority state variable
NODE 15 P3 REG ; priority state variable

NODE 6 B0 REG ; PCI bus state variable
NODE 10 B1 REG ; PCI bus state variable

NODE 34 T0 REG ; time-out counter bit
NODE 36 T1 REG ; time-out counter bit
NODE 38 T2 REG ; time-out counter bit
**Backplane 0015 Interface**

**NODE 49**

T3 REG ; time-out counter bit

**NODE 37**

TIMEOUT REG ; time-out node

**NODE 53**

RST1n REG ; reset synchronizing register

**NODE 54**

RESETn REG ; synchronous reset

**NODE 14**

AGENTQUIT COMB ; granted bus agent deasserts bus request

**STRING sH**

`(/P3 * /P2 * /P1 * /P0 )`

**STRING sA**

`(/P3 * /P2 * /P1 * P0 )`

**STRING sB**

`(/P3 * /P2 * P1 * /P0 )`

**STRING sC**

`(/P3 * P2 * /P1 * /P0 )`

**STRING sD**

`( P3 * /P2 * /P1 * /P0 )`

**STRING sGRANTED**

`( B1 * /B0 )`

**STRING sPARKED**

`( B1 * B0 )`

**STRING sOFF**

`( /B1 * B0 )`

**STRING sACTIVE**

`( B1 * /B0 )`

**EQUATIONS**

HGNTn.CLKF = CLK HGNTn.SETF = GND

AGNTn.CLKF = CLK AGNTn.SETF = GND

BGNTn.CLKF = CLK BGNTn.SETF = GND

CGNTn.CLKF = CLK CGNTn.SETF = GND

DGNTn.CLKF = CLK DGNTn.SETF = GND

P0.CLKF = CLK P0.RSTF = /ASYNCRSTn

P1.CLKF = CLK P1.RSTF = /ASYNCRSTn

P2.CLKF = CLK P2.RSTF = /ASYNCRSTn

P3.CLKF = CLK P3.RSTF = /ASYNCRSTn

B0.CLKF = CLK B0.SETF = GND

B1.CLKF = CLK B1.SETF = GND

T0.CLKF = CLK T0.SETF = GND

T1.CLKF = CLK T1.SETF = GND

T2.CLKF = CLK T2.SETF = GND

T3.CLKF = CLK T3.SETF = GND

RST1n.CLKF = CLK RST1n.SETF = GND

RESETn.CLKF = CLK RESETn.SETF = GND

TIMEOUT.CLKF = CLK TIMEOUT.SETF = GND

;****************************;
;** KEEP UNUSED INPUTS IN FILE **;
;****************************;

DUMMY1 = TRDYn * STOPn * LOCKn

;****************************;
;** SYNCRONOUS RESET GENERATION **;
;****************************;
Backplane 0015 Interface

RST1n := ASYNCRSTn
RESETn := RST1n

;*******************************************************************************;
;** BUS GRANT TIMEOUT EQUATIONS **;
;*******************************************************************************;

T0 := RESETn * sGRANTED * IRDYN * /T0
  + RESETn * sGRANTED * IRDYN * T1 * /T0

T1 := RESETn * sGRANTED * IRDYN * /T1 * T0
  + RESETn * sGRANTED * IRDYN * T2 * /T0
  + RESETn * sGRANTED * IRDYN * T2 * T1 * T0

T2 := RESETn * sGRANTED * IRDYN * /T2 * T1 * T0
  + RESETn * sGRANTED * IRDYN * T2 * /T0
  + RESETn * sGRANTED * IRDYN * T2 * T1 * T0
  + RESETn * sGRANTED * IRDYN * T2 * T1 * /T0

T3 := RESETn * sGRANTED * IRDYN * /T3 * T2 * T1 * T0
  + RESETn * sGRANTED * IRDYN * T3 * /T1 * T0
  + RESETn * sGRANTED * IRDYN * T3 * T2 * /T0
  + RESETn * sGRANTED * IRDYN * T3 * T2 * T1 * T0
  + RESETn * sGRANTED * IRDYN * T3 * T2 * T1 * /T0
  + RESETn * sGRANTED * IRDYN * T3 * T2 * /T0

TIMEOUT := T3 * T2 * T1 * T0

AGENTQUIT := /HGNTn * HREQn
  + /AGNTn * AREQn
  + /BGNTn * BREQn
  + /CGNTn * CREQn
  + /DGNTn * DREQn

;*******************************************************************************;
;** BUS GRANT EQUATIONS **;
;*******************************************************************************;

/HGNTn := sH * sGRANTED * /TIMEOUT
  + sH * sACTIVE
  + sH * sSPARKED

/AGNTn := sA * sGRANTED * /TIMEOUT
  + sA * sACTIVE
  + sA * sSPARKED

/BGNTn := sB * sGRANTED * /TIMEOUT
  + sB * sACTIVE
  + sB * sSPARKED

/CGNTn := sC * sGRANTED * /TIMEOUT
  + sC * sACTIVE
  + sC * sSPARKED

/DGNTn := sD * sGRANTED * /TIMEOUT
  + sD * sACTIVE
  + sD * sSPARKED

;*******************************************************************************;
;** PCI BUS STATE MACHINE **;
;*******************************************************************************;
**STATE TRANSITIONS**

sPARKED := (any REQn=0) -> sOFF  
+ (all REQn=1) -> sPARKED

sOFF := (FRAMEn=1 & IRDYn=1) -> sGRANTED
+ (FRAMEn=0) -> sACTIVE
+ (FRAMEn=1 & IRDYn=0) -> sACTIVE

sGRANTED := (FRAMEn=0 & ANY GNT=0) -> sOFF
+ (FRAMEn=0 & ALL GNT=1) -> sACTIVE
+ (FRAMEn=1 & TIMEOUT=0) -> sGRANTED
+ (FRAMEn=1 & TIMEOUT=1) -> sPARKED
+ (AGENTQUIT=1) & ALL REQ=1) -> sPARKED
+ (AGENTQUIT=1) & ANY REQ=0) -> sOFF

sACTIVE := (any REQn=0 & FRAMEn=1) -> sGRANTED
+ (all REQn=1 & FRAMEn=1) -> sPARKED
+ (FRAMEn=0) -> sACTIVE

B0 := sPARKED * AREQn * BREQn * CREQn * DREQn * HREQn
+ sPARKED * (/AREQn + /BREQn + /CREQn + /DREQn + /HREQn)
+ sGRANTED * /FRAMEn * (/AGENTn + /BGNTn + /CGNTn + /DGNTn + /HGNn)
+ sGRANTED * FRAMEn * TIMEOUT
+ sGRANTED * AGENTQUIT
+ sACTIVE * FRAMEn * AREQn * BREQn * CREQn * DREQn * HREQn
+ /RESETn

B1 := sPARKED * AREQn * BREQn * CREQn * DREQn * HREQn
+ sOFF * /FRAMEn
+ sOFF * FRAMEn * /IRDYn
+ sGRANTED * FRAMEn * TIMEOUT
+ sGRANTED * /FRAMEn * AGNTn * BGNn * CGNTn * DGNTn * HGNn
+ sGRANTED * AGENTQUIT * AREQn * BREQn * CREQn * DREQn * HREQn
+ sACTIVE * /FRAMEn
+ sACTIVE * FRAMEn * AREQn * BREQn * CREQn * DREQn * HREQn
+ /RESETn

;******************************;
** PRIORITY STATE MACHINE **;
;******************************;

STATE
MOORE_MACHINE
DEFAULT_BRANCH HOLD_STATE

;** STATE ASSIGNMENT EQUATIONS **;

H = /P3 * /P2 * /P1 * /P0 ; This state assignment results
A = /P3 * /P2 * /P1 * P0 ; in 11 product terms per variable.
B = /P3 * /P2 * P1 * /P0 ; The conventional assignment
C = /P3 * P2 * /P1 * /P0 ; results in 14 or more product
D = P3 * /P2 * /P1 * /P0 ; terms for some variables.

/** STATE TRANSITION EQUATIONS **;

H := H_TO_A -> A
    + H_TO_B -> B
    + H_TO_C -> C
    + H_TO_D -> D

A := A_TO_B -> B
    + A_TO_C -> C
    + A_TO_D -> D
    + A_TO_H -> H

B := B_TO_C -> C
    + B_TO_D -> D
    + B_TO_H -> H
    + B_TO_A -> A

C := C_TO_D -> D
    + C_TO_H -> H
    + C_TO_A -> A
    + C_TO_B -> B

D := D_TO_H -> H
    + D_TO_A -> A
    + D_TO_B -> B
    + D_TO_C -> C

CONDITIONS

/** STATE CONDITION EQUATIONS **;

H_TO_A = /AREQn * sOFF
H_TO_B = AREQn * /BREQn * sOFF
H_TO_C = AREQn * BREQn * /CREQn * sOFF
H_TO_D = AREQn * BREQn * CREQn * /DREQn * sOFF

A_TO_B = /BREQn * sOFF
A_TO_C = BREQn * /CREQn * sOFF
A_TO_D = BREQn * CREQn * /DREQn * sOFF
A_TO_H = BREQn * CREQn * DREQn * /HREQn * sOFF

B_TO_C = /CREQn * sOFF
B_TO_D = CREQn * /DREQn * sOFF
B_TO_H = \text{CREQ}_n \times \text{DREQ}_n \times \overline{\text{HREQ}}_n \times \text{sOFF}

B_TO_A = \text{CREQ}_n \times \text{DREQ}_n \times \text{HREQ}_n \times \overline{\text{AREQ}}_n \times \text{sOFF}

C_TO_D = \overline{\text{DREQ}}_n \times \text{sOFF}

C_TO_H = \text{DREQ}_n \times \overline{\text{HREQ}}_n \times \text{sOFF}

C_TO_A = \text{DREQ}_n \times \text{HREQ}_n \times \overline{\text{AREQ}}_n \times \text{sOFF}

C_TO_B = \text{DREQ}_n \times \text{HREQ}_n \times \text{AREQ}_n \times \overline{\text{BREQ}}_n \times \text{sOFF}

D_TO_H = \overline{\text{HREQ}}_n \times \text{sOFF}

D_TO_A = \text{HREQ}_n \times \overline{\text{AREQ}}_n \times \text{sOFF}

D_TO_B = \text{HREQ}_n \times \text{AREQ}_n \times \overline{\text{BREQ}}_n \times \text{sOFF}

D_TO_C = \text{HREQ}_n \times \text{AREQ}_n \times \overline{\text{BREQ}}_n \times \overline{\text{CREQ}}_n \times \text{sOFF}